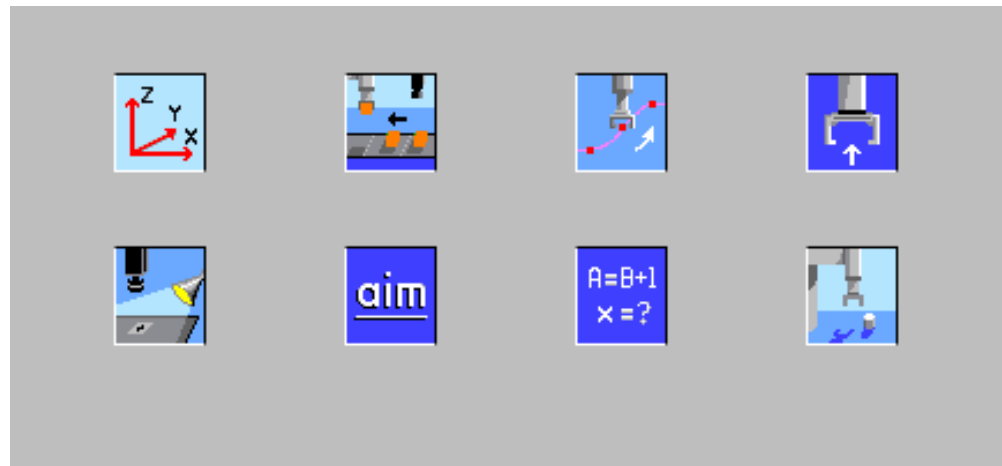

MotionWare

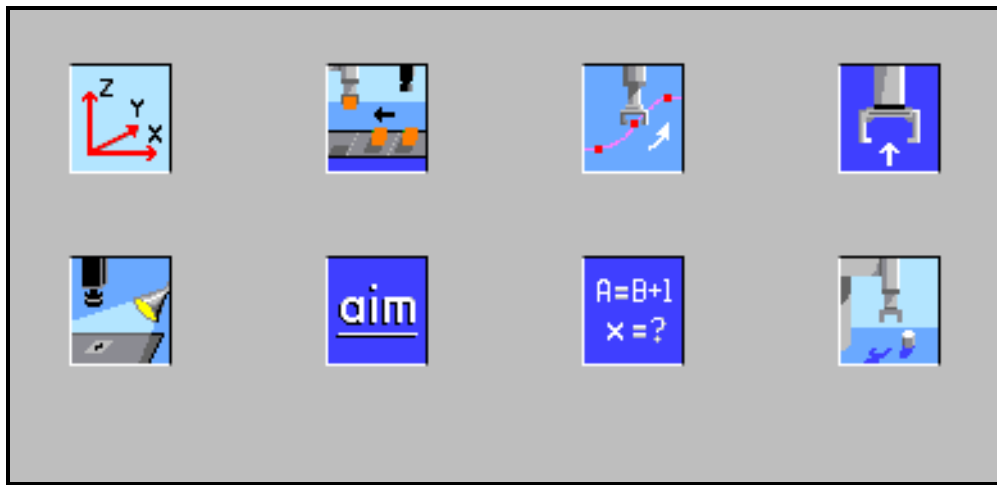
User's Guide



Version 3.1

MotionWare

User's Guide



Version 3.1

Part Number 00713-01630, Rev. A

January 1997



150 Rose Orchard Way • San Jose, CA 95134 • USA • Phone (408) 432-0888 • Fax (408) 432-8707

Otto-Hahn-Strasse 23 • 44227 Dortmund • Germany • Phone (49) 231.75.89.40 • Fax(49) 231.75.89.450

41, rue du Saule Trapu • 91300 • Massy • France • Phone (33) 1.69.19.16.16 • Fax (33) 1.69.32.04.62

1-2, Aza Nakahara Mitsuya-Cho • Toyohashi, Aichi-Ken • 441-31 • Japan • (81) 532.65.2391 • Fax (81) 532.65.2390

The information contained herein is the property of Adept Technology, Inc. and shall not be reproduced in whole or in part without prior written approval of Adept Technology, Inc. The information herein is subject to change without notice and should not be construed as a commitment by Adept Technology, Inc. This manual is periodically reviewed and revised.

Adept Technology, Inc. assumes no responsibility for any errors or omissions in this document. Critical evaluation of this manual by the user is welcomed. Your comments assist us in preparation of future documentation. A form is provided at the back of the book for submitting your comments.

Copyright © 1992, 1997 by Adept Technology, Inc. All rights reserved.

The Adept logo is a registered trademark of Adept Technology, Inc.

Adept, AdeptOne, AdeptOne-MV, AdeptThree, AdeptThree-MV, PackOne, PackOne-MV, HyperDrive, Adept 550, Adept 550 CleanRoom, Adept 1850, Adept 1850XP, A-Series, S-Series, Adept MC, Adept CC, Adept IC, Adept OC, Adept MV, AdeptVision, AIM, VisionWare, AdeptMotion, MotionWare, PalletWare, FlexFeedWare, AdeptNet, AdeptFTP, AdeptNFS, AdeptTCP/IP, AdeptForce, AdeptModules, AdeptWindows, AdeptWindows PC, AdeptWindows DDE, AdeptWindows Off-line Editor, and V⁺ are trademarks of Adept Technology, Inc.

Any trademarks from other companies used in this publication are the property of those respective companies.

Printed in the United States of America

Table Of Contents

Read Me First!	1
Compatibility	1
What Is MotionWare?	1
What Is New in MotionWare, Version 3.1?	1
What Do I Already Need to Know?	2
What Should I Already Have Done?	2
Software Installation	2
Do I Have to Read All the Manuals?	2
How Do I Use MotionWare?	4
How Can I Get Help?	6
Within the Continental United States	6
Service Calls	6
Application Questions	6
Applications Internet E-Mail Address	6
Training Information	6
Within Europe	7
France	7
Outside Continental United States or Europe	7
Adept Bulletin Board Service	7

PART ONE BASIC MOTIONWARE

1 Getting Started	11
1.1 Loading and Executing MotionWare	12
1.2 Using the Basic Interface	13
Working With the Pointing Device	13
Working in Windows	13
Clicking and Dragging	14
Opening a Window (Making a Menu Selection)	14
Selecting a Window	14
Moving a Window	14
Scrolling Within a Window	15
Sizing a Window	15
Closing a Window	15
Main Menu Bar	15
Window Menu Bar	15
1.3 Using MotionWare Menu Pages	15

Standard Menu Features	15
About This Manual	16
Menu Selection Example	17
Keyboard Conventions	18
Selecting, Choosing, and Pressing	18
Selecting	18
Choosing	18
Pressing	19
1.4 Trace Enable/Disable	19
Trace Enable/Disable Options	19
1.5 Editing Records	20
Selecting (Highlighting)	20
Active/Inactive	20
The Typing Cursor	20
Creating New Records	21
Copying and Deleting Records	21
Selecting Different Records	21
Exiting Record Editing	22
Record-Editing Menu Bars and Quick Keys	23
1.6 Additional Record-Creation Aids	23
1.7 Access Levels	23
Assigning Access Levels	24
Creating User Accounts	24
Editing User Accounts	24
Logging On the System	25
Logging Off the System	25
2 Basic Concepts	27
2.1 Robot Locations	28
Location Values	28
2.2 Motion Parameters	30
Straight-Line versus Joint-Interpolated Motion	31
Speed	31
Rotational Speed	31
Speed Units	31
Location Precision	32
Robot Acceleration and Deceleration	32
Profile	32
Approaching and Departing a Location	32
Arm Configuration	33
Using the Manual Control Pendant (MCP) to Teach Locations	35
2.3 The MotionWare Databases	35
Location	35
Path	36
Frame	36
Tool	36

Conveyor	36
Force.	36
Variable.	37
Vision and Camera.	37
2.4 Preview: MOVE Statement.	37
3 MotionWare Resource Modules	39
3.1 What Is a Resource Module?	40
Loading and Selecting Resource Modules	41
Resource Module Directory	42
3.2 Managing Resource Modules.	42
3.3 Resource Module Options	42
Module Resources	45
Global Databases.	47
Dual-Vision Systems and AIM	47
4 Sequence Creation and Execution.	49
4.1 Introduction	50
4.2 Sequences and Resource Modules	50
Control Sequences	50
Creating a Sequence	50
The Sequence Index.	50
4.3 Editing a Sequence.	52
Selecting a Statement.	52
Statement Components	52
Completing a Statement Argument.	53
Adding a Sequence Line.	53
Completing Optional Statement Arguments	53
Optional Clauses	53
Completing Constant Arguments	54
Adding Comments to a Sequence	54
Removing Sequence Lines.	54
Cutting, Copying, and Pasting Statement Lines	55
Copying Statement Lines Between Sequences	55
Sequence Editor Special Keys	56
Finding and Finding/Changing Statement Arguments	59
4.4 What Are the Sources for Statement Arguments?	60
Avoiding Trouble With Statement Arguments.	61
4.5 The Statement Tree	62
Statement Tree Options	63
Source Records	63
4.6 Task Control Panel	64
Task Control Panel Options	64
4.7 Debugging Sequences	66
Task Control Debug Options	67

Controlling the Robot	68
4.8 Using AIM Tasks	68
How Do I Decide Which Tasks to Run?	69
How Do I Start a Task?	69
4.9 Master Control Panel	70
4.10 Editing Database Records During Execution	71
Activating Single-Step Execution	72
5 Location Databases	73
5.1 Editing Locations	74
Location Record Options	74
Example Strategy Routines	78
Optimized Moves	78
5.2 Using Pallets	80
Pallet Parameters	81
How Rows and Columns Are Determined	81
6 Variable Database	83
6.1 Introduction	84
6.2 Accessing the Variable Database	84
Variable Database Record Options	84
A Note on Data Conversion	87
SELECT_VAR_REC Statement	87
7 Paths	89
7.1 What Are Paths?	90
7.2 Creating Paths	91
Path Menu Page Options	92
7.3 Editing Path Segments	94
Path Segment Menu Page Options	94
8 Reference Frames	97
8.1 What Is a Reference Frame?	98
8.2 Creating Reference Frames	100
Frame Menu Options	100
8.3 Tool Transformations	102
Creating a Tool Transformation	103
Current Tool	104
Tools and Statements	104
9 Using Digital I/O	105
9.1 Introduction	106
9.2 Installing Digital I/O	106
Displaying the State of Digital I/O Signals	106

9.3 Digital I/O Signal Numbering	107
Soft Signals.	107
Special Signals	108
Negative Signal Numbers.	108
9.4 Using Digital I/O With MotionWare	108
Setting the State of Digital Signals	108
Waiting for a Digital Signal	108
Digital I/O and Conveyor Belts	109
Digital I/O and Locations.	109
OK_SIGNAL	109
Control Statements.	109
9.5 Using Digital I/O Statements.	109
WAIT_FOR Statement	109
WAIT_UNTIL Statement.	110
IO Statement.	110
IO_LIST Statement.	111
IO_LIST_VAR Statement	111
10 Using the I/O Server	113
10.1 Introduction	114
10.2 Installing the I/O Server	114
10.3 I/O Protocol Routines	114
10.4 Using I/O Server Statements	114
IO_OPEN Statement	115
IO_CLOSE Statement	115
IO_READ Statement.	115
IO_WRITE Statement.	116
10.5 Example I/O Server Sequence	117
10.6 Example Serial Driver Sequence	117
11 Force Sensing Module	119
11.1 Introduction	120
AdeptForce versus Digital Force Sensors	120
Enabling Force Sensing in AIM	120
Configuring V ⁺ for Digital Force Sensors	120
11.2 Modes of Operation	121
Guarded Mode	121
Protect Mode	123
11.3 Operator Interface (AdeptForce Only)	124
Force Sensor Status and Control Display.	124
Force Signature Display Page	125
11.4 Force Process Database (AdeptForce and Digital Sensor)	126
Force Process Records	127
Force Process Routines	131
11.5 Example Application	132

Part Acquisition	133
Part Insertion	134
12 Using Conveyor Systems	139
12.1 Indexing vs. Tracking Conveyors	140
Conveyor Database, Frame Database, and Location Database	140
Conveyor Terminology	140
Conveyor Objects	141
12.2 Conveyor Overview	141
12.3 Indexing Conveyor Records	142
12.4 Using an Indexing Conveyor	143
Indexing Conveyor: Runtime Operation	144
12.5 Conveyors and Pallets	144
12.6 Tracking Conveyor Strategy	144
12.7 Tracking Conveyor Frame Records	146
12.8 Tracking Conveyor Options	147
12.9 Using a Tracking Conveyor	150
Single-Access Operation	151
Multiple-Access Option	151
Variable-Access Option	152
12.10 Calibrating a Tracking Conveyor	152
Belt Calibration Procedure	153
Calibrating Multiple Robots with a Single Conveyor	154
Belt Calibration and External Files	154
12.11 Conveyor Queue Summary	155
12.12 Conveyor Statements	157
GET_FRAME Statement	157
RELEASE_FRAME Statement	160
CHECK_OBJECT Statement	160
MATCH_OBJECTS Statement	161
RESUME_CONVEYOR Statement	161
12.13 Conveyor Tracking Example	162
Create a Resource Module	163
Create a Conveyor Record	163
Create a Frame Record	164
Create a Location Record	165
Calibrate the Tracking Conveyor	165
Install an Object Sensor	166
Create a Sequence	166
13 MotionWare Statements	167
13.1 Introduction	168
13.2 General MotionWare Statements	169
MOVE Statement	169
How MOVE Uses the Databases	170

Things to Remember	172
MOVE_REL Statement	172
DRIVE Statement	173
SET_FRAME Statement	173
Things to Remember	174
SET_TOOL Statement	174
STOP_ROBOT Statement	174
WAIT_UNTL_ROBOT Statement	174
RESET_PALLET	176
MESSAGE	176
SET_RESPONSE	176
13.3 General AIM Statements	177
Using Expressions in Statements	177
Conditional and Looping Structures	178
CALL Statement	178
IF Statement	178
END Statement	179
ELSE Statement	179
FOR Statement	180
REPEAT Statement	181
WHILE Statement	181
EXIT_LOOP Statement	182
NEXT_LOOP Statement	182
CASE Structure (Includes VALUE and ANY Statements)	182
GOTO Statement	183
SET Statement	183
SETS Statement	183
RETURN Statement	185
13.4 Sequence Control Statements	185
LOAD Statement	185
PAUSE_TASK Statement	185
RESUME_TASK Statement	186
SELECT_TASK Statement	186
START_TASK Statement	186
STOP_TASK Statement	186
UNLOAD Statement	187
WAIT Statement	187
WAIT_VAR Statement	187

PART TWO MOTIONWARE WITH VISION

14 MotionWare and Vision 191

14.1 Getting Started With Vision	192
14.2 Camera Calibration	192
Camera Calibration Records	193

Camera Calibration: Basic Steps	193
14.3 Calibration Records	194
14.4 Calibration Options	196
Vision Z-Axis Direction	196
Fixed-Mount Calibration Option	197
Fixed-Mount: General Case	197
Downward-Looking With Vacuum Gripper	199
Arm-Mounted Camera Calibration	200
14.5 Calibration-Process Window.	202
14.6 MotionWare With Vision Statements	203
INSPECT	203
PICTURE Statement.	204
Vision Inspection Using a Fixed-Mount Camera.	204
Vision Inspection Using an Arm-Mounted Camera.	206
14.7 REFINE_TOOL Statement	207
Sequence of Operations.	207
Vision Refinement for the Gripped Part	208
14.8 REFINE_FRAME Statement.	210
Using REFINE_FRAME	210
Statement Syntax and Arguments	211
Sequence of Operations	211
Records Required by REFINE_FRAME	211
Fiducial Location Records	211
Vision Tool Record	212
Optional Records for REFINE_FRAME	213
Additional Considerations.	214
15 Using Conveyor Systems With Vision	215
15.1 Vision With Indexing and Tracking Conveyors	216
15.2 Vision and Indexing Conveyors	219
15.3 Vision-Based Tracking Conveyor Options	222
Vision and Tracking Conveyors	224
15.4 Advanced Conveyor Operations.	224
16 MotionWare Utilities	227
16.1 The File Manager	228
File Manager Options	228
16.2 The Backup and Restore Utility	231
The Backup Menu Page	231
Backup Definitions	231
Creating a Backup Definition	232
Restoring Files	234
Restore Options	236

A	Installing MotionWare.	237
A.1	Installing the Software	238
	Create a Subdirectory for the AIM Files	238
	Copy the AIM Files to the Hard Drive	238
	Upgrade the Operating System	239
A.2	Initialization Databases	240
	Initialization Options	240
A.3	Autostarting MotionWare	244
B	Menu Bars and Quick Keys.	245
B.1	Main Menu Bar	246
B.2	Record Editing Menu Bar and Quick Keys	248
B.3	Vision Window Menu Bar	251
B.4	Sequence Editing Menu Bar and Quick Keys	253
	Index.	257

List of Figures

Figure 1-1	Start-up Screen	12
Figure 1-2	Monitor Window Feature Identification.	13
Figure 1-3	AdeptWindows PC Monitor Window Feature Identification	14
Figure 1-4	MotionWare Menu Features	16
Figure 1-5	Trace Enable/Disable.	19
Figure 1-6	Searching for a Database Value	22
Figure 2-1	Robot Location Characteristics	28
Figure 2-2	World Coordinate System	29
Figure 2-3	Yaw, Pitch, and Roll.	30
Figure 2-4	Location With Approach and Depart Heights.	33
Figure 2-5	Left/Right Robot Arm Configuration Option	34
Figure 2-6	Setting Robot Arm Configuration	34
Figure 3-1	Example Resource Modules	41
Figure 3-2	New Module Menu Page	42
Figure 3-3	Edit Module Components Menu Page	44
Figure 3-4	Select New Module Component Type	45
Figure 3-5	Edit Module Component	46
Figure 4-1	Sequence Index.	51
Figure 4-2	Sequence Editor	52
Figure 4-3	Find and Change Windows	59
Figure 4-4	Sample Statement Tree.	62
Figure 4-5	Task Control Panel	64
Figure 4-6	Task Control Panel With Debug Options	66
Figure 4-7	Master Control Panel	70
Figure 5-1	Location Database Menu Page	74
Figure 5-2	Using Strategy Sequences	75
Figure 5-3	Location Strategies Menu Page.	76
Figure 5-4	Location Configuration Menu Page	76
Figure 5-5	Elements of an Optimized Move	79
Figure 5-6	Optimized Approach/Move.	79
Figure 5-7	Pallet Parameters	80
Figure 6-1	Variable Database Menu Page	84
Figure 7-1	Example of an "Along" Path.	91
Figure 7-2	Path Menu Page	92
Figure 7-3	Path Segment Menu Page.	94
Figure 8-1	World Coordinate System	98
Figure 8-2	Locations Relative to a Reference Frame	99
Figure 8-3	Frame Menu Page	100

Figure 8-4	Tool Transformation Example	102
Figure 8-5	Tool Record	103
Figure 8-6	Display/Change Tool Menu Page	104
Figure 9-1	Digital I/O Control.	107
Figure 11-1	Force Sensor Status and Control Menu Page	124
Figure 11-2	Force Signature Display Menu Page	125
Figure 11-3	Force Process Index Page.	126
Figure 11-4	Location Strategies Page.	127
Figure 11-5	Force Process Database Record.	128
Figure 11-6	Edit Trip Conditions Page	129
Figure 11-7	“Strategy” Page for Peg Feeder Location	133
Figure 11-8	Part Acquisition Example, Force Process Record 1 of 1.	134
Figure 11-9	Part Insertion Example, Force Process Record 1 of 3.	135
Figure 11-10	Part Insertion Example, Force Process Record 2 of 3.	136
Figure 11-11	Search Parameters	137
Figure 11-12	Part Insertion Example, Force Process Record 3 of 3.	138
Figure 12-1	Conveyor Terminology.	141
Figure 12-2	Indexing Conveyor.	142
Figure 12-3	Tracking Conveyor.	147
Figure 12-4	Define Conveyor Frame Record	149
Figure 12-5	Conveyor Belt Status Display	155
Figure 12-6	Conveyor Belt Object Status	156
Figure 12-7	Example Conveyor Tracking Setup	162
Figure 14-1	Vision Calibration Window	194
Figure 14-2	Fixed-Mount Camera Calibration.	198
Figure 14-3	Arm-Mounted Camera Calibration	200
Figure 14-4	Calibration-Process Window	202
Figure 14-5	Picture Record Motion Options (Fixed-Mount Camera).	205
Figure 14-6	Picture Record Motion Options (Arm-Mounted Camera).	206
Figure 14-7	REFINE_FRAME Example	210
Figure 15-1	Conveyors and Vision	217
Figure 15-2	Indexing Vision Conveyor	219
Figure 15-3	Frame Record for Indexing Conveyor With Vision.	220
Figure 15-4	Vision-Based Tracking Conveyor.	222
Figure 15-5	Frame Record for Tracking Conveyor With Vision.	223
Figure 16-1	File Manager Window	228
Figure 16-2	Backup Selection	231
Figure 16-3	List of Backup Definitions.	232
Figure 16-4	Backup Definition.	232
Figure 16-5	List of Backup Savesets.	235
Figure 16-6	Restore Menu Page.	236

List of Tables

Table 1	Manuals You Should Read or Review	3
Table 2	Manuals Used for Optional Modules or Custom Programming	3
Table 1-1	Conventions Used in This Manual for Menu Instruction Lines.	16
Table 1-2	Typographic Conventions Used in This Manual	18
Table 4-1	Statement Argument Sources	60
Table 11-1	Summary of Force System Operation Modes	121
Table 11-2	Standard Force Process Routines.	131
Table 12-1	Waiting for Status Messages	155
Table 13-1	MOVE Statement Databases.	170
Table 15-1	Records That Return Frame Results ("Frame Class" Records).	218
Table A-1	Options in the Baseline File (BASEINI.DB)	240
Table A-2	Options in the Robot Module File (ROBINI.DB).	242
Table A-3	Options in the MotionWare Initialization File (MOWINI.DB).	243
Table A-4	Options in the User File (USERINI.DB)	243
Table B-1	Record Editing Quick Keys	250
Table B-2	Sequence Editing Quick Keys	254

Read Me First!

Compatibility

This manual is for use with MotionWare, version 3.1, in conjunction with V⁺ version 11.2 or later. The MotionWare software version number is printed on the back of each MotionWare disk. If the version of your software is not 3.1, contact Adept at the phone number listed at the end of this chapter. See Appendix A for details on the V⁺ version.

What Is MotionWare?

MotionWare is a module designed for the Adept AIM system. MotionWare allows you to create sophisticated robot workcell implementations without requiring low-level programming.

What Is New in MotionWare, Version 3.1?

MotionWare, version 3.1, includes the following new features:

- AdeptWindows PC support. This is an optional interface for the Adept AIM system and a Microsoft Windows®-compatible personal computer (PC). Using AdeptWindows PC, you can access and operate the AIM system (running on an MV controller) from your PC. See the *AIM 3.1 Release Notes* for details.
- Dynamic Data Exchange (DDE) server capability. This feature allows you to share information with DDE-compatible applications on your PC. See the *AIM 3.1 Release Notes* for details.
- FlexFeedWare support. This option allows you to install, program, and operate a Flexible Feeder from the MotionWare module. See the *AIM 3.1 Release Notes* for details.
- AIM Advisor capability. This feature provides interactive on-line instructions for installing and configuring new hardware, and defining and creating new part records. This feature currently supports only the Flexible Feeder with FlexFeedWare. See the *AIM 3.1 Release Notes* for details.
- I/O Server capability. This feature provides tools for handling I/O operations such as reading/writing to disk files, serial ports, and a network connection. It also provides a uniform interface to shared resources such as datalog files, bar-code readers, and RF tag readers without having to write custom AIM statements. See Chapter 10 for details.
- Enhanced variable database. This enhancement supports string variables and access to general AIM databases and vision results.

What Do I Already Need to Know?

You should be familiar with your robot and its capabilities. In particular, you should know:

- How to power up the controller.
- How to power up and perform start-up calibration of the robot.
- How to power up and adjust any other devices in the workcell.
- The safety requirements of your robot and any other devices that operate in the workcell.



WARNING: When run carelessly or by inexperienced operators, robots and other motion devices can severely injure personnel and cause equipment damage.

- Basic operation and use of the optional manual control pendant (MCP) if you will be using it.

What Should I Already Have Done?

Hardware that will be controlled by MotionWare should already be installed and tested. If you will be using hardware in the following list, it should be installed before proceeding.

- The robot and any of the following options that you will be using:
 - Fifth axis
 - Force sensing module
- The controller and any of the following options:
 - Digital I/O
 - Cameras and strobes (if vision is used)
- Cell equipment, including:
 - Part feeders
 - Conveyors
 - Connections between the cell equipment and the digital I/O system

Software Installation

Appendix A covers installing the MotionWare system on your hard disk drive (a hard disk drive is required to run AIM).

Do I Have to Read All the Manuals?

MotionWare comes with a complete set of reference material that allows you not only to use MotionWare, but also to customize MotionWare at the programming (V+) level and write your own vision and robot programming applications. The manuals you should

read are listed in Table 1. The manuals you can ignore unless you are customizing AIM are listed in Table 2.

Table 1
Manuals You Should Read or Review

Manual	Material Covered
<i>MotionWare User's Guide</i>	How to use the MotionWare software.
Robot user's guide	Basic capabilities of your robot, as well as how to perform the start-up procedures.
Controller user's guide	Basics of using the controller. If you are using digital I/O, pay particular attention to the requirements for initializing and installing the digital I/O hardware.
<i>Instructions for Adept Utility Programs</i>	Instructions for running the different Adept utility programs. Depending on which options you use, you may have to run different Adept utility programs. Keep the manual handy for instructions on any utility programs you may have to run.
<i>VisionWare User's Guide</i>	(Vision users only.) Description of the vision processes that can be used in conjunction with MotionWare. If you are using the vision option, you need to read this manual carefully.
<i>AIM 3.1 Release Notes</i>	User and reference information for Adept AIM, version 3.1. It covers everything that is not detailed in the <i>MotionWare User's Guide</i> .

Table 2
Manuals Used for Optional Modules or Custom Programming

Manual	Material Covered
AIM reference manuals	AIM database structures and the routines used to drive the AIM system. There is either a manual or a section of a manual for the base AIM system and each AIM module that you have. If you are going to use MotionWare as delivered, you can ignore this material.
<i>AIM Customizer's Reference Guide</i>	The basics of customizing the AIM software. Only system customizers need to read this manual.
<i>V+ Language Reference Guide</i>	The operating system and language that all V+ and AIM programs are written in. Unless you are writing custom V+ or AIM code, you can ignore this material.
<i>AdeptVision Reference Guide</i>	(Vision users only.) Details of the vision enhancements to the V+ language. This reference manual is a companion to the <i>V+ Language Reference Guide</i> .
<i>AdeptVision VME User's Guide</i>	(Vision users only.) The "how to" material for Adept's vision system. Unless you are programming custom vision applications, you can ignore this manual, although a quick review of the manual will give you an idea of how the vision system works.

Table 2
Manuals Used for Optional Modules or Custom Programming (Continued)

Manual	Material Covered
<i>AIM CAD Data Translator User's Guide</i>	(CAD data translator only.) Using the CAD data translator to automatically create database records from CAD data.
<i>AdeptForce VME User's Guide</i>	The "how to" material for Adept's six-axis force sensing package. If you are using a simple digital force sensor with MotionWare, you can ignore this material.

How Do I Use MotionWare?

Before you can begin using MotionWare to control your workcell, complete the following steps:

- Install the controller (see the controller user's guide).
- Install the robot (see the robot user's guide).
- Install any safety devices needed to prevent injuries during workcell operation.
- Install any other devices that will work in conjunction with the robot.
- If you are using digital I/O:
 - Install the I/O modules in the controller (see the controller user's guide).
 - Connect the I/O modules to the devices in the workcell.
- If you are using vision:
 - Install the cameras (and strobes if used).
 - Calibrate the cameras (see "Camera Calibration" on page 192).

If you have not used any other AIM modules, read Chapter 1 for details on using the MotionWare window and menu system.

Read Chapter 1 for a general overview of MotionWare. Several elementary concepts that are covered in this chapter must be understood to effectively use MotionWare.

If you have not used other AIM modules, Chapter 4 describes creating statements (the basic unit of a MotionWare workcell implementation) and sequences (a completed MotionWare workcell implementation). It also describes how to execute a sequence. You must know how to create and execute sequences before you can use MotionWare to execute robot operations. Chapter 13, Chapter 14, and Chapter 15 discuss programming statements used in MotionWare.

MotionWare implementations that you create will include moving to a location and executing a gripper action at that location. Basic MotionWare operations require you to complete the following actions:

1. Create a resource module that defines the databases and other resources you will use for your applications. See Chapter 3.
2. Create a location database. Creating a location database involves specifying the robot locations, motion parameters for the location, and gripper action at the location. Chapter 5 covers building location databases. Section 2.1 describes a typical MotionWare location.

3. Create a sequence that specifies in what order the robot should move to these locations. See Chapter 4.

In addition to the required actions, MotionWare allows you to take several optional actions to more effectively and accurately control the robot.

These actions are:

1. Create paths (or safe corridors) along which the robot can move without damaging equipment in the workcell or other parts on the assembly.

A path is a series of locations. Once the robot enters a path, it should be able to move from one location in the path to another without causing damage. Chapter 7 describes the use and creation of paths.

2. Create reference frames.

If part locations will not always be in the same location in the workcell, reference frames simplify updating of the locations. Chapter 8 shows you how to create and use reference frames. Chapter 8 also describes tool transformations, which are used to correct for the difference between the center of the robot tool flange and the actual center of the gripper.

3. Define and calibrate conveyor belts that supply or remove parts (or assemblies).

If your system has the vision option, the vision system can:

- Inspect the parts picked up by the robot (Chapter 14).
- Locate parts on an indexing conveyor belt (Chapter 15).
- Track randomly oriented parts on moving conveyors (Chapter 15).

How Can I Get Help?

Within the Continental United States

Adept Technology maintains a Customer Service Center at its headquarters in San Jose, CA. The phone numbers are:

Service Calls

(800) 232-3378 (24 hours a day, 7 days a week)
(408) 433-9462 FAX

NOTE: When calling with a controller-related question, please have the serial number of the controller. If your system includes an Adept robot, also have the serial number of the robot. The serial numbers can be determined by using the ID command (see the *V⁺ Operating System User's Guide*).

Application Questions

If you have an application question, you can contact the Adept Applications Engineering Support Center for your region:

Adept Office	Phone #, Hours	Region
San Jose, CA	Voice (408)434-5033 Fax (408)434-6248 8:00 A.M. – 5:00 P.M. PST	Western Region States: LA, AR, MO, TX, OK, KS, NE, CO, WY, MT, NM, AZ, UT, NV, ID, WA, OR, CA
Cincinnati, OH	Voice (513)792-0266 Fax (513)792-0274 8:00 A.M. – 5:00 P.M. EST	Midwestern Region States: MI, OH, West PA, West NY, IN, KY, TN, AL, MS, IL, WI, IA, MN, ND, SD
Southbury, CT	Voice (203)264-0564 Fax (203)264-5114 8:00 A.M. – 5:00 P.M. EST	Eastern Region States: ME, NH, VT, MA, CT, RI, East NY, East PA, NJ, DE, MD, VA, WV, NC, SC, GA, FL

Applications Internet E-Mail Address

If you have access to the Internet, you can send applications questions by e-mail to:
applications@adept.com

This method also enables you to attach a file, such as a portion of V⁺ program code, to your message.

Training Information

For information regarding Adept Training Courses in the USA, please call (408) 434-5024.

Within Europe

Adept Technology maintains a Customer Service Center in Dortmund, Germany. The phone numbers are:

(49) 231 / 75 89 40 from within Europe (Monday to Friday, 8:00 A.M. to 5:00 P.M.)

(49) 231/75 89 450 FAX

France

For customers in France, Adept Technology maintains a Customer Service Center in Paris, France. The phone numbers are:

(33) 1 69 19 16 16 (Monday to Friday, 8:30 A.M. to 5:30 P.M., CET)

(33) 1 69 32 04 62 FAX

Outside Continental United States or Europe

For service calls, application questions, and training information, call the Adept Customer Service Center in San Jose, California, USA:

1 (408) 434-5000

1 (408) 433-9462 FAX (service requests)

1 (408) 434-6248 FAX (application questions)

Adept Bulletin Board Service

Adept maintains a bulletin board service (BBS) for customer use. The BBS number is (203) 264-5590. Application utility programs and helpful hints are posted to this bulletin board, and users may also post their own hints and application notes. There is no charge for access to the Adept bulletin board. (You will, of course, incur normal long-distance phone charges for the call to the BBS.) The first time you call the BBS, you will be able to set up an account right from the BBS. For questions about the BBS, call (800) 232-3378.

MOTIONWARE USER'S GUIDE

PART ONE

BASIC MOTIONWARE

Getting Started

1

1.1 Loading and Executing MotionWare	12
1.2 Using the Basic Interface	13
Working With the Pointing Device	13
Working in Windows	13
Clicking and Dragging	14
Opening a Window (Making a Menu Selection)	14
Selecting a Window	14
Moving a Window	14
Scrolling Within a Window	15
Sizing a Window	15
Closing a Window	15
Main Menu Bar	15
Window Menu Bar	15
1.3 Using MotionWare Menu Pages	15
Standard Menu Features	15
About This Manual	16
Menu Selection Example	17
Keyboard Conventions	18
Selecting, Choosing, and Pressing	18
Selecting	18
Choosing	18
Pressing	19
1.4 Trace Enable/Disable	19
Trace Enable/Disable Options	19
1.5 Editing Records	20
Selecting (Highlighting)	20
Active/Inactive	20
The Typing Cursor	20
Creating New Records	21
Copying and Deleting Records	21
Selecting Different Records	21
Exiting Record Editing	22
Record-Editing Menu Bars and Quick Keys	23
1.6 Additional Record-Creation Aids	23
1.7 Access Levels	23
Assigning Access Levels	24
Creating User Accounts	24
Editing User Accounts	24
Logging On the System	25
Logging Off the System	25

1.1 Loading and Executing MotionWare

If MotionWare has not already been copied to the hard disk drive, see Appendix A. Once MotionWare has been copied to the hard disk drive, use the following procedure to load AIM into system memory and start up MotionWare.

1. Turn on the Adept controller and monitor. The system will start up and display a series of messages on the monitor window. When the screen looks similar to the screen shown in Figure 1-1, you are ready to start MotionWare.

Copyright (c) 1984-1996 by Adept Technology, Inc. All rights reserved.

Software: 12.0 83-11C0 (Edit A3, 10-Apr-1996)

Controller: 3302-773 0

Processor 1: 0.0 2-7 8Mb

Robot 1: 100-0 0-0 8

Robot 2: 100-0 0-0 8

Vision 1: 12.0 500 1 1.25Mb

28-May-96 15:11:22

Figure 1-1
Start-up Screen

2. Make the AIM subdirectory (or whichever subdirectory you copied MotionWare to) the default directory.

Type:

default disk = c:\aim and press the ENTER key.

3. Load and execute the software.

Type:

load lmow and press the ENTER key.

For a single-robot system, type:

comm lmow and press the ENTER key.

For a two-robot system, type:

comm lmow2 and press the ENTER key¹

¹ As delivered, MotionWare requires the V+ Extensions option to use more than one robot.

1.2 Using the Basic Interface

Working With the Pointing Device

The pointing device controls the position of the cursor arrow on the monitor screen. The pointing device can be used for selecting windows, moving windows on the screen, and opening applications. A standard Microsoft compatible mouse uses the center mouse button. The trackball on the standard Adept keyboard uses only the left button.

Working in Windows

Figure 1-2 shows the features of a standard Adept window when the Adept video monitor and keyboard are attached to the controller. Figure 1-3 shows the features of the Adept window if you are using the controller with a PC and AdeptWindows PC. The callouts shown in both figures are described in the rest of this section.

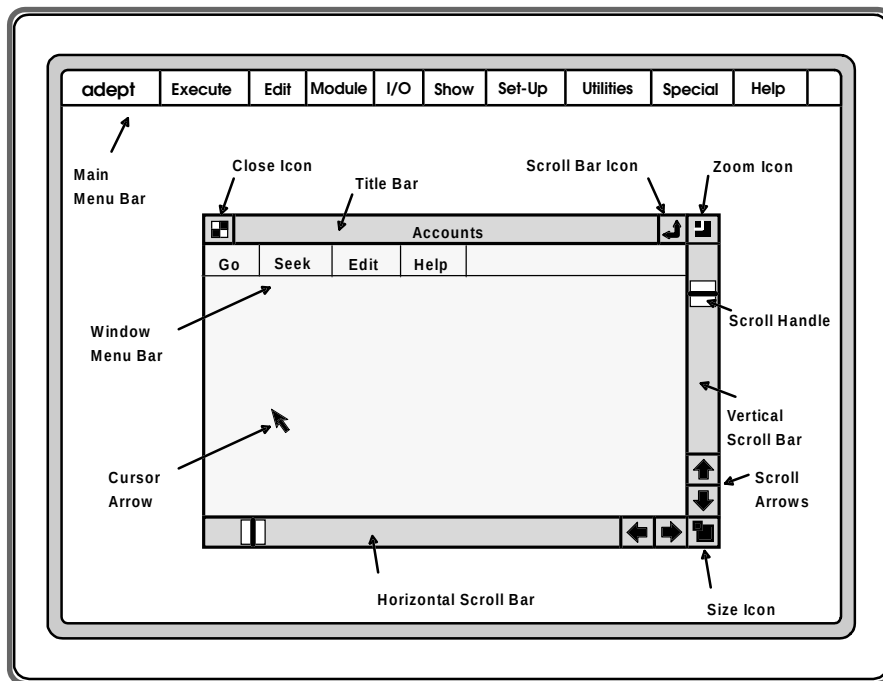


Figure 1-2
Monitor Window Feature Identification

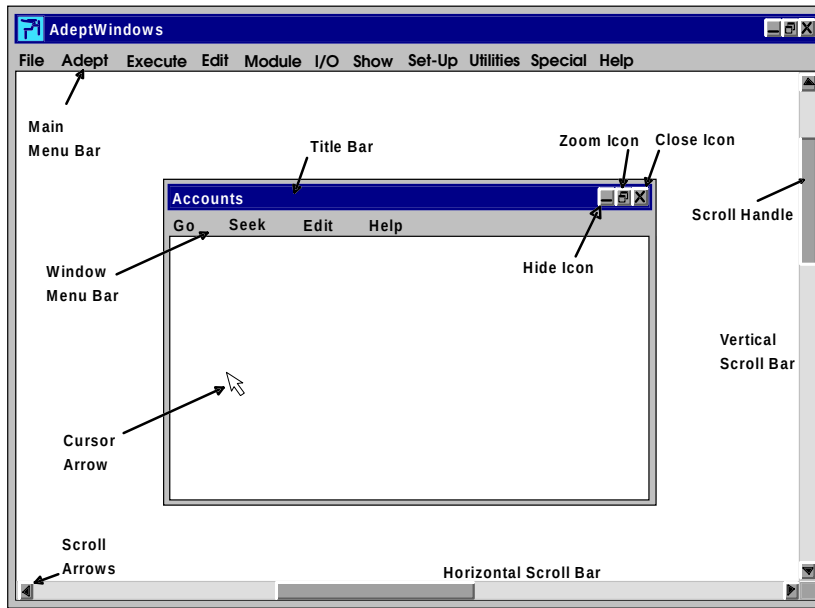


Figure 1-3
AdeptWindows PC Monitor Window Feature Identification

Clicking and Dragging

To click an item, position the cursor arrow (using the mouse or trackball) over the item; press and release the button.

To drag an item, position the cursor arrow over the item, press and hold down the button, then drag the item to a new location and release the button.

Opening a Window (Making a Menu Selection)

1. Position the pointer on the **adept** logo at the top left corner of the screen.
2. Press and hold down the button to display a pull-down menu. Drag the pointer until the command that you want to open is selected.
3. Release the button to open the window.

Multiple windows can be open at one time, and windows can overlap each other. When windows overlap, the active (selected) window appears on top of any other open windows.

Selecting a Window

You can work only in the currently selected window. To select a window, click anywhere inside the window. The title bar will change from white to dark blue when the window is selected. You can also make any open window the currently selected window by choosing that window from the pull-down menu under the **adept** logo.

Moving a Window

Click the window title bar and drag the window to a new location.

Scrolling Within a Window

The scroll bars can be turned on and off by clicking the scroll-bar icon. The arrows at the end of the scroll bar allow you to scroll through a window one line at a time. Drag the scroll handles or click anywhere in the scroll bar to scroll quickly to any part of the window. (Scroll handles are not displayed when the window has been expanded to its full size.)

Sizing a Window

You can change the size of a window by clicking the size icon (lower right corner) and dragging it to a new size. Another way to change the size of a window is to click the zoom icon to toggle the window between its maximum size and its current reduced size.

Closing a Window

To close a window, click the close icon in the upper left corner. Closing the vision or monitor window does not terminate the process that is running in that window. The process will continue to run in the background. For example, if you are formatting a disk, closing the monitor window will not stop the format process.

Main Menu Bar

This menu bar is displayed whenever MotionWare is being run. Use of the main menu bar options is summarized in Appendix B.

Window Menu Bar

Each open window has its own menu bar with pull-down options. Menu bars are specific to the window in which they appear. Menu bars for MotionWare windows are summarized in Appendix B.

1.3 Using MotionWare Menu Pages

Standard Menu Features

Screens (windows) displayed by MotionWare are called menu pages. Motion applications are built by making selections from the MotionWare menu pages. Menu page features are common to many menu pages. The appearance of each menu page feature tells you what will happen when you select a given feature. Figure 1-4 describes the menu page features.

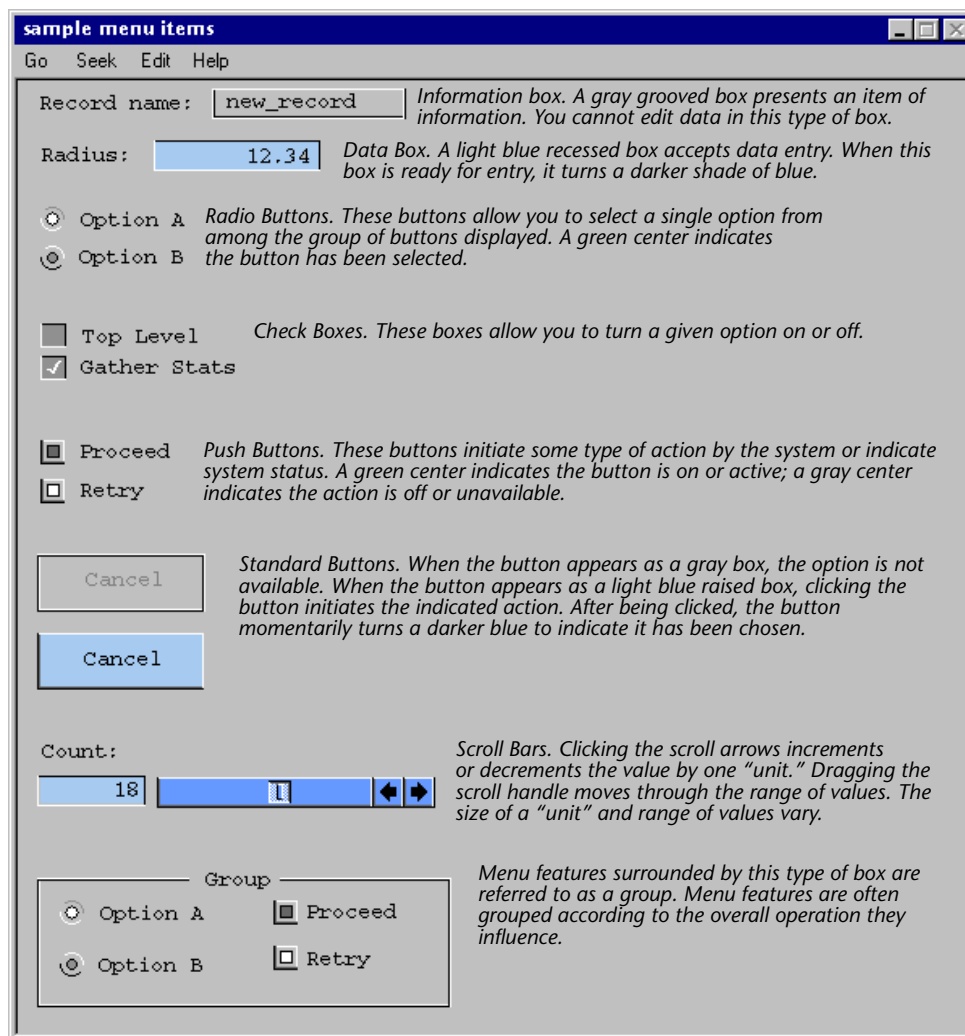


Figure 1-4
MotionWare Menu Features

About This Manual

To facilitate understanding and use of menu bars and menu page selections, this manual uses the following conventions:

Table 1-1
Conventions Used in This Manual for Menu Instruction Lines

Symbol or Appearance	Action
Edit	Select (click and hold down mouse button) the menu-bar item with the same title.
Part	Choose this option from the pull-down options presented after a menu-bar item has been selected.
➡	Move to the next action in the instruction sequence.

Table 1-1
Conventions Used in This Manual for Menu Instruction Lines (Continued)

Symbol or Appearance	Action
Goto	Choose (click) the button with the same name.
☐ Arc	Select the radio button with the same name.
☒ Relative	Select the check box with the same name.
<i>enter "hi"</i>	Actions such as entering data or text or double-clicking an item are shown in italics.
*	This symbol calls your attention to an important point.

Menu Selection Example

The instruction line

Edit ➡ **Part Type** ➡ **Edit** ➡ **New Record** ➡ *Enter "my.rec" in the name data box* ➡ ☒ **Search enabled** ➡ **Done**

indicates that you should perform the following actions:

1. Select **Edit** from the main menu bar.
2. Choose **Part Type** from the resulting pull-down.
3. Select **Edit** from the newly displayed window menu bar.
4. Choose **New Record** from the resulting pull-down menu.
5. Type **my.rec** into the name data box and press ENTER.
6. Select the **Search enabled** check box.
7. Choose the **Done** button.

Table 1-2
Typographic Conventions Used in This Manual

Appearance	Representation
<i>italics</i>	Indicates new terms and other emphasized words.
SMALL CAPS	Indicates a physical key or button than you must press, such as the ENTER key.
bold	In a typing instruction, type anything that you see in this font exactly as it appears. What you type is shown in lowercase letters unless it must be typed in uppercase letters to work properly. For the V ⁺ system to process your typing, you must conclude your entry by pressing the ENTER or RETURN key.
regular monospace	Used for monitor displays, code examples, non placeholder terms in formal syntax definitions, and case-sensitive words required for a UNIX-like setting associated with AdeptNet.
Menu bold	In text discussion of menu pages, this type style is used for menu items that you choose and window items that do not have initial capital letters in all principal words.

Keyboard Conventions

Key combinations appear in the following format:

Notation	Meaning
KEY1+KEY2	A plus sign(+) between keys means that you must press the keys at the same time. For example, "Press CTRL+Z" means that you press CTRL and hold it down while you press Z.

Selecting, Choosing, and Pressing

The terms select, choose, and press (as used in this manual) have different and specific meanings.

Selecting

This implies either marking a window item or highlighting it. For example, you may mark a check box with a check. Selecting alone does not initiate an action.

You can use a combination of keyboard and pointer-device techniques for selecting.

Choosing

This implies carrying out an action. For example, choosing a menu item may open a window or execute a command. You can also initiate an action by choosing a command button (a push button or a standard button). Often, you will select an item and then choose a button to initiate the desired action.

You can use a combination of keyboard and pointer-device techniques for choosing.

Pressing

This refers to physical buttons or keys such as those found on a keyboard controller or manual control pendant. For example, you press the **SAVE** key on a keyboard or the **PROG SET** button on a Manual Control Pendant. By contrast, you select or choose a button appearing in a displayed MotionWare window.

1.4 Trace Enable/Disable

MotionWare allows you to display “trace” messages in the Monitor window. (You can view the monitor window by choosing it from the pull-down menu under the **adept** logo.) Trace messages show what events are being processed by the menu drivers and what sequence statements are being executed.

To display trace messages:

Special ➔ Trace Enable/Disable

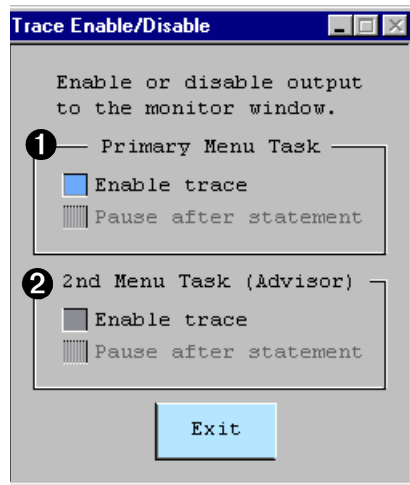


Figure 1-5
Trace Enable/Disable

Trace Enable/Disable Options

- ❶ This group is used to display the primary menu task.
Select ☒ **Enable trace** to display the primary menu task trace messages in the Monitor window.
Select ☒ **Pause after statement** to pause after each control sequence statement is executed.

- ❷ This group is used to display the second menu (Advisor) task.

NOTE: Advisor tasks apply only to the Flexible FeedWare module.

- Select ☒ **Enable trace** to display the primary menu task trace messages in the Monitor window.
Select ☒ **Pause after statement** to pause after each control sequence statement is executed.

1.5 Editing Records

Selecting (Highlighting)

MotionWare operations require you to select a menu item or a name in a scrolling list. To select a data box or an entry in a scrolling list, click once on the data box or the list item. The background of the selected item will turn a dark blue. After a data box is selected, text entered at the keyboard will be stored in the data box; existing text in the data box will be deleted.

Use this procedure to use your mouse to select standard buttons, push buttons, radio buttons, and check boxes:

- Press the TAB key to select successive items on a menu page.
- SHIFT-TAB moves backward through the menu items.
- Holding down the SHIFT key and pressing one of the cursor movement keys (key marked by ←, →, ↑, or ↓ arrows) will move through the menu items in an up/down or left/right manner.
- When a standard button, radio button, push button, or check box is selected, pressing the ENTER key is the same as clicking that button.
- When a standard button is selected, it turns dark blue.
- When a radio button is selected, the outer ring turns light blue.
- When a push button is selected, the outer square turns light blue.
- A selected check box has a blue background.

Active/Inactive

In many cases, menu items or menu page options will not be active for the operation that you are currently performing. When this situation occurs, the item is inactive and will appear dimmed (displayed as a light shade of gray).

The Typing Cursor

The typing cursor (a solid black square) indicates where keystrokes will appear when entered. Characters typed at the keyboard will always be entered to the left of the typing cursor. The BACKSPACE key deletes the character to the left of the cursor. The DELETE key deletes the character under the cursor.

To set the typing cursor in a data box:

- Click once on the data box to select the data box. (Characters entered from the keyboard will replace characters in the data box.)
- Select a data box and click where you want to anchor the typing cursor. Characters entered from the keyboard will be inserted to the right of the cursor. (Since many data boxes will respond to being double-clicked, this second click must be made after a short pause.)
- The cursor can be moved within the data box by pressing the cursor-arrow keys. Existing text will not be affected.

Creating New Records

To create a new record, pull down the Edit menu and choose a database such as Frame or Location. The database menu page is displayed.

Press the NEW key (F2), or:

Edit ➡ New Record

Copying and Deleting Records

To remove a record from the database, press the CUT key (SHIFT+F9) or:

Edit ➡ Cut Record

The currently displayed record will be cut from the database and placed in the copy buffer.

To copy a record, press the COPY key (F9) or:

Edit ➡ Copy Record

Only one record is held in the copy buffer at a time. If a second record is cut or copied, the first record will be lost.

To paste or cut a record, press the PASTE key (F10) or:

Edit ➡ Paste Record

NOTE: When a copied record is pasted to a database, two records will exist with the same name in the database. The new record should immediately be renamed to prevent duplicate record names.

When a sequence is run, record names are used in the linking process; duplicate record names may create problems. (An error message will be generated whenever a database has two records with the same name. One of the records should be deleted or renamed.)

Selecting Different Records

MotionWare provides several methods for moving between records in a database. The **Seek** menu item allows the following options:

Option	Definition
Index	Displays a scrolling list of all records in the database. Select the desired record and choose Goto to go to that record.
First (HOME)	Goes to the first record in the database.
Previous (PAGE UP)	Goes to the record immediately preceding the current one.
Next (PAGE DOWN)	Goes to the next record in the database.
Last (END)	Goes to the last record in the database.

Option	Definition
Find (FIND-F7)	Displays the String window (see Figure 1-4). A data box must be selected or an information box must be selected: When searching for a string, selecting ☒ Match any substring (see Figure 1-6) causes MotionWare to find the specified text in any position ("and" would be found in "sandy"). To find only the word "and", do not select the check box. When searching for a numeric value, enter a tolerance; any value that is within the tolerance will be found. For example, if 5 is the search string and the tolerance is 0.1, any values between 4.9 and 5.1 will be found. Choosing First searches all records in the database. Choosing Next searches from the next record to the last record in the database. The search will be performed only on the selected field.
Repeat Find (REPEAT-F8)	Repeats the last completed find operation. This is useful for locating duplicate record names.

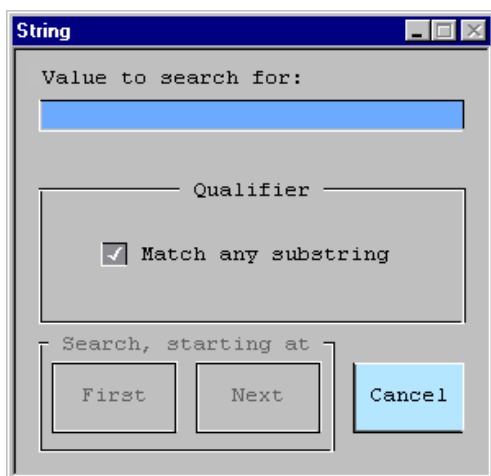


Figure 1-6
Searching for a Database Value

Exiting Record Editing

To exit the database editor, perform any of the following options:

- Select a different item from the main menu bar.
- Click the close icon in the upper left corner of the window.
- Press the EXIT key (F4).
- **Go ➡ Exit.**

Record-Editing Menu Bars and Quick Keys

The record-editing menu bar is summarized in Appendix B. Quick keys can be used in lieu of menu selections; they are summarized also in Appendix B.

1.6 Additional Record-Creation Aids

In addition to the menus, MotionWare provides several other aids to creating vision records. The following table discusses additional record-creation aids.

Aid	Description
Double-clicking	Double-clicking any empty data box that requires a MotionWare record will display a list of records that can be placed in the field. When the list is displayed, double-click a record name to retrieve that record. Choose Retrieve to place the record name in the data box, New to create a new record, or GoTo to go to that record. A record name entered into a field on a menu page is referred to as a source record. The field in which the record is placed is referred to as a source field. Double clicking a data box that already contains a record name for its value will take you to that record.
	This symbol (yellow on the screen) indicates that the record name you have entered does not exist, is the wrong type, or is not complete. It must be created or changed before the operation will be valid.

1.7 Access Levels

There are five access levels in MotionWare:

Level 0 (Minimum Access) allows a user to log on to the system.

Level 1 (System Operator) allows a user to run existing sequences.

Level 2 (Hardware Maintenance) allows access for equipment repair.

Level 3 (Manufacturing Engineer) allows creation of new sequences and vision records.

Level 4 (System Customizer) allows complete access to the system, including the creation of new user accounts.

The menu and menu page options that are not available to the currently logged-on level will be dimmed (shown in a lighter shade of gray).

When MotionWare starts, default access is level 4, which provides complete access to the system. If you are going to use the system security option, set it up before any further operations are undertaken.

To change the start-up access level:

Setup ➔ Initialization Data ➔ double-click BASEINI

Seek ➔ Index ➔ double-click 'access level, initial'

In the Value data box, set the desired start-up access level.



CAUTION: After changing this value, ensure that at least one user account with access level 3 or higher remains in the accounts database. Otherwise, you will lock yourself out of level 3 and 4 menu items.

Assigning Access Levels

Your system administrator should set up initial access levels using the following procedure:

1. Create a new user account for the system administrator, and set the access level to 4.
2. Create user accounts for other system users.

Creating User Accounts

Only persons with access level 4 can set up or edit user accounts.

To add a user:

1. **Special ➔ Edit User Accounts ➔ Edit ➔ New Record**
2. Enter the user's name in the User data box.
3. Enter the user's password in the Password data box.¹
4. Click the radio button corresponding to the access level for the user.
5. Exit by **Go ➔ Exit**, by clicking the close icon, or by pressing the EXIT key (F4).

Editing User Accounts

To edit a user account:

Special ➔ Edit User Accounts

To change a user's access level:

Seek ➔ Index ➔ select the user name ➔ Goto

Click the new access level the user is to have.

¹ Passwords can be up to six characters. They must begin with an alpha character and can include numeric characters, underscores, and periods.

To delete a user account:

Seek ➡ Index ➡ *select the user name* ➡ Goto

Edit ➡ Cut Record

Logging On the System

To log on as the current user:

1. **Utilities ➡ Set Access Level**
2. Enter your user name in the User Name data box.
3. After a user name has been entered, the Password data box is displayed. Enter your password in the Password data box.
4. If log on is successful, the radio buttons next to the access levels you are authorized for will be selected and you can select a level to work at.
5. Exit by pressing the EXIT key (F4).

Logging Off the System

1. **Utilities ➡ Set Access Level**
2. Select ☒ **Minimum access**.
3. Exit by pressing the EXIT key (F4).

NOTE: After a user has logged off, another log on must be successfully completed before any further MotionWare operations can be completed.

Basic Concepts **2**

2.1 Robot Locations	28
Location Values	28
2.2 Motion Parameters	30
Straight-Line versus Joint-Interpolated Motion	31
Speed	31
Rotational Speed	31
Speed Units	31
Location Precision	32
Robot Acceleration and Deceleration	32
Profile	32
Approaching and Departing a Location	32
Arm Configuration.	33
Using the Manual Control Pendant (MCP) to Teach Locations	35
2.3 The MotionWare Databases	35
Location	35
Path	36
Frame	36
Tool.	36
Conveyor	36
Force.	36
Variable.	37
Vision and Camera	37
2.4 Preview: MOVE Statement.	37

2.1 Robot Locations

This section describes how to specify a location that the robot can move to and the parameters that can be set to alter how a robot moves to that location.

A robot location is a predefined position at which a robot will acquire or place a part or move through en route to acquiring or placing a part. Locations have controllable characteristics that produce smooth, efficient robot operation.

Figure 2-1 shows a typical robot location screen. The text in this figure discusses each of the location characteristics that can be specified. You will see screens with options similar to Figure 2-1 in several of the following chapters. You should have a good understanding of each of the location options before you move on to the rest of this manual.

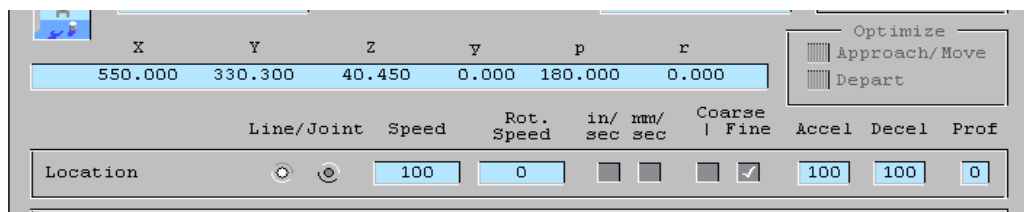


Figure 2-1
Robot Location Characteristics

Location Values

At the top of Figure 2-1 is a data box labeled **X Y Z y p r**. This data box stores the Cartesian coordinate values for a specific robot location and the orientation of the *end effector*¹ at that location. These six values tell a robot exactly where it should go and what position its end effector should be in when it gets to that location.

The X, Y, and Z values correspond to a point in Cartesian space relative to the base frame of the robot.² This Cartesian space is called the *world coordinate system*. Figure 2-2 shows the world coordinate system for a SCARA³ robot and for a Cartesian robot.

¹ Any gripper, tooling, or equipment that is mounted to the robot tool flange.

² Later on, we will describe how to make locations relative to reference frames other than the robot base.

³ Refers to a robot with the link and joint configuration shown at the left of Figure 2-2.

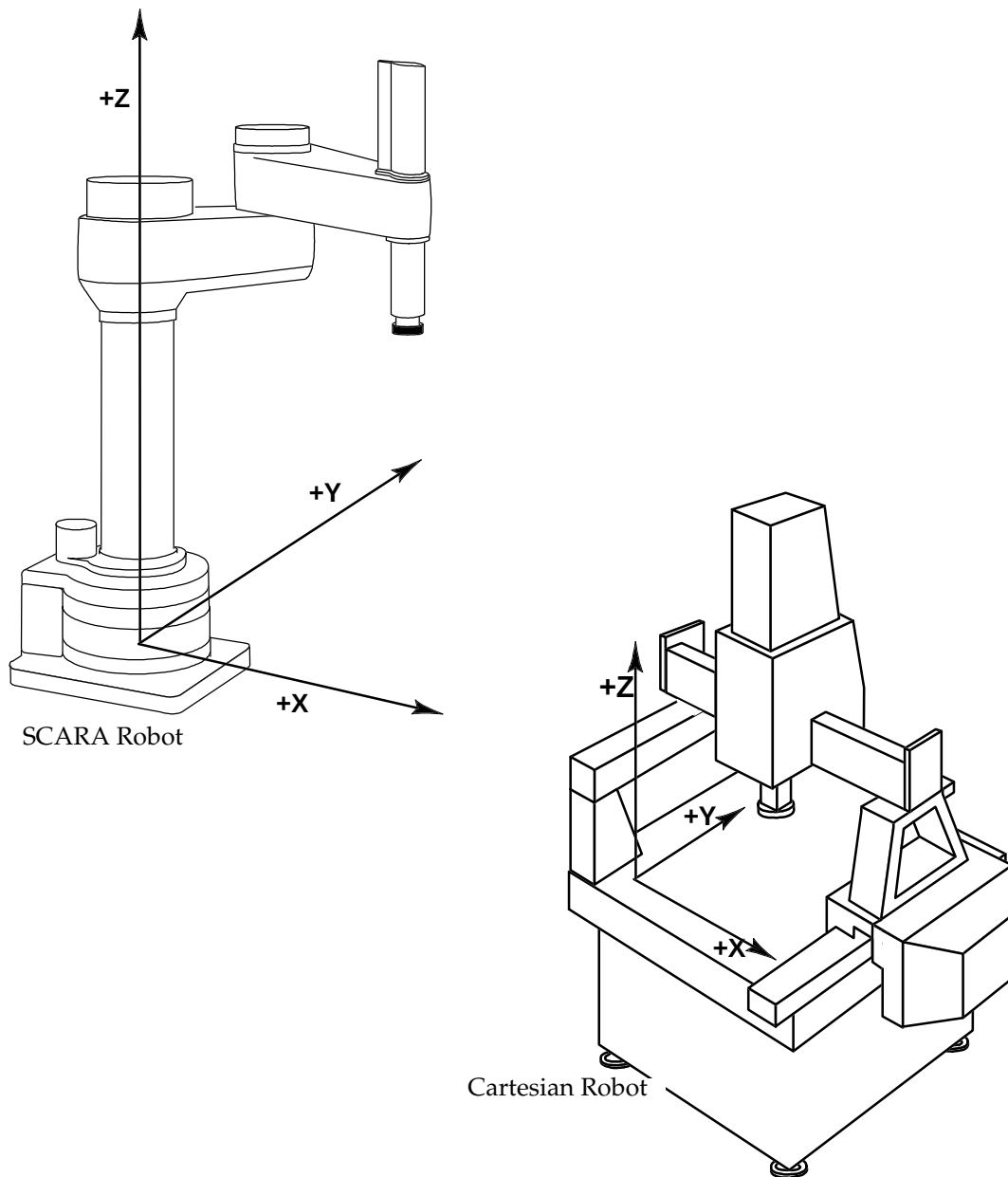


Figure 2-2
World Coordinate System

The y, p, and r (yaw, pitch, and roll) values correspond to the orientation of the end effector at that point.¹ Figure 2-3 shows the meaning of these location components. Notice that with a pitch of 180° (pointing straight down), yaw and roll describe the same motion (although the positive and negative directions are opposite).

¹ Unlike X, Y, and Z values, y, p, and r are ordered values. That is, yaw must be applied first, then pitch is applied, and, finally, roll is applied to arrive at the correct orientation.

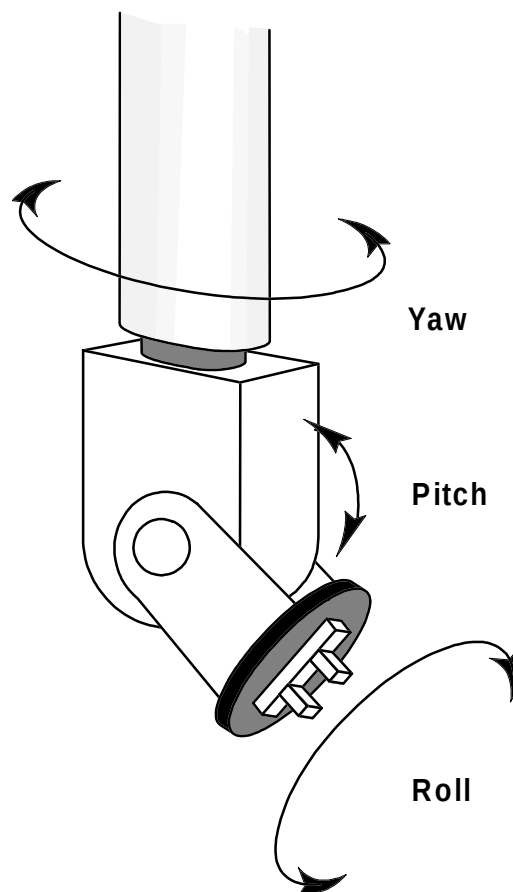


Figure 2-3
Yaw, Pitch, and Roll

Locations values can be either entered manually¹ or generated and stored automatically by MotionWare. To store a new location, you place the robot at the correct location and orientation and choose **Here**. A window is displayed prompting you to verify the recording. Location values are automatically calculated (based on the current robot location) and stored in the proper record.

If you have the optional Adept manual control pendant (MCP), choosing **Teach** will allow you to use the MCP to move the robot and teach locations. In general, you do not have to be concerned with the values of a location variable. If you create a location that is incorrect, it can be changed by placing the robot in the correct location and choosing either **Here** or **Teach**.

2.2 Motion Parameters

The following parameters affect the way the robot moves to a location. Motion parameters are specified for each location.

¹ The optional AIM CAD data translator will create locations from CAD data.

Straight-Line versus Joint-Interpolated Motion

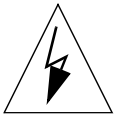
Adept SCARA robots can move from one location to another using one of these two types of motion. The fastest and least processing-intensive motion is joint interpolated. In this type of motion, the controller ignores the movement of the tool flange and concentrates on moving the robot joints to the proper values. Thus, the actual path the end effector takes is not necessarily a straight line.

The second type of motion is straight line. If this type of motion is specified, the robot end effector will move in a straight line from one location to another. Straight-line motion is useful for applying beads of sealant or ensuring that the end effector traces a predictable path. Unless you have a specific reason for selecting straight-line motion, select joint-interpolated motion. These motions are selected with the Line/Joint radio buttons.

Speed

Robot speed is usually specified as a percentage of normal speed, not as an absolute velocity. (See “Speed Units” on page 31 for directions on setting absolute speed.) Speed is set in two places. Overall speed for an entire application is set from the control panel (see “Task Control Panel” on page 64). The speed for a single robot motion is set in the Speed data box of each individual location. This individual speed setting will increase or decrease the normal speed for the move to the location by the percentage specified.

After the location is reached, robot speed will return to its previously set value. The result of setting the two speed variables is cumulative. If 50 is specified at the control panel and 50 is specified with the location, the actual speed will be 25% of normal for the move to that location.



WARNING: To avoid possible injury and damage to equipment when you are first developing an implementation, set the overall speed to a value less than 25%. In some mechanisms, speed can be raised higher than the normal speed by entering a value greater than 100 (the maximum is 200). Speed should be raised above 100% only with extreme caution.

Rotational Speed

If a tool with a large offset is attached to a robot, the robot’s joint and flange speeds can be very large when rotations about the tool tip occur during straight-line motions. Rotational speed controls the maximum tool-rotation speed during straight-line motions. The rotational speed for a single robot motion is set in the Rot. Speed data box of each individual location.

If a rotational speed factor is specified, it is interpreted as a percentage of maximum Cartesian rotation speed to be used during straight-line motions. If rotational speed is not specified, one of the following results occurs:

- If speed units also are not specified, the rotational speed is set to the value of the speed factor.
- If speed units are specified, the rotational speed is not changed.

Speed Units

If straight-line motion is selected, speed can be set to an absolute value by selecting either ☒ **mm/sec** or ☒ **in/sec**. The value in the Speed data box will now be interpreted as an absolute value, and the robot will move at the indicated speed. The speed set at the

control panel scales this value, so for the speed to be accurate, the speed at the control panel must be set to 100.

Location Precision

When a robot moves to a location, it actually makes several moves, each of which is a closer approximation of the exact location. You can control the precision with which the robot moves to a location using the Coarse/Fine setting. If coarse is selected, the robot will spend less time attempting to reach the exact location. In many cases, this setting will prove completely adequate, and robot cycle times can be improved by setting location precision to coarse.

If neither option is selected, the robot will move smoothly through the location without stopping. This type of motion is referred to as continuous-path motion. Continuous-path motion is useful for paths and approach and depart locations where the robot does not actually stop to acquire or place a part. Approaches and departs are described later in this section. Paths are described in Chapter 7.

Robot Acceleration and Deceleration

You can control the rate at which the robot reaches its designated speed and stops using the acceleration and deceleration parameters. Like speed, acceleration and deceleration rates are specified as a percentage of the normal acceleration and deceleration rates. To make the robot start or stop less abruptly, set the values in the Accel and Decel data boxes to values less than 100.

These parameters are commonly used with vacuum grippers. Abrupt stops with a vacuum gripper may cause the part being held to shift on the gripper. This problem could be solved by lowering robot speed. However, overall cycle time would be increased. An alternative is to slow down the acceleration and deceleration rates so the part does not shift on the gripper during start or stop. The robot can still move at the maximum designated speed for other movements.

Profile

Defined profiles set the maximum rate of change of acceleration and deceleration. You specify the profile by typing an integer from 0 to 8 in the Prof data box. If you specify a profile that has not been defined, profile 0 is used. For non-Adept robots, see the *AdeptMotion VME User's Guide* for details on defining acceleration profiles. For Adept robots, see the robot user's guide for details on available profiles.

Approaching and Departing a Location

Approach and depart heights can be specified for many locations. A typical location with approach and depart heights is specified in a screen similar to Figure 2-4.

	Line/Joint	Speed	Rot. Speed	in/sec	mm/sec	Coarse	Fine	Accel	Decel	Prof
Location	☐ ☐	100	0	☐	☐	☐	☒	100	100	0
Approach	10 ☐ ☐	10	0	☐	☐	☐	☐	100	100	0
Depart	5 ☐ ☐	25	0	☐	☐	☐	☐	100	100	0

Figure 2-4
Location With Approach and Depart Heights

When approach and depart heights are specified, the robot moves in three distinct motions. In the first motion, the robot moves to a location directly above the specified location. The height above the location is specified in the Approach data box. In the second motion, the robot moves to the actual location and the gripper is activated.¹ In the third motion, the robot moves to a point directly above the location. The height of the depart motion is specified in the Depart data box.

Approach and depart heights are used to make sure that the robot approaches and departs from a location without running into any other parts in the assembly being built (or any other obstructions in the robot envelope). Approaches and departs are always parallel to the Z-axis of the tool coordinate system. Notice that all the motion parameters that apply to a move to a location also can be applied to approach and depart moves. This allows you to move at optimum speed to the approach height above a location, then move more slowly when actually acquiring or placing the part.

Arm Configuration

One final motion parameter that you can control is the configuration of the arm when a location is moved to. Configuration options apply only to specific types of robots. For example, the left/right option applies to SCARA-type robots such as the AdeptOne and AdeptThree. Figure 2-5 shows how a SCARA robot can reach a point with the elbow pointing in different directions. When the configuration menu page is displayed, only the configuration options that are appropriate for the specified robot are active. See the device module description for the relevant options for your non-Adept motion devices.

¹ When the robot moves to an approach location, the gripper is not necessarily in the correct orientation. The gripper is properly oriented during the move to the actual location.

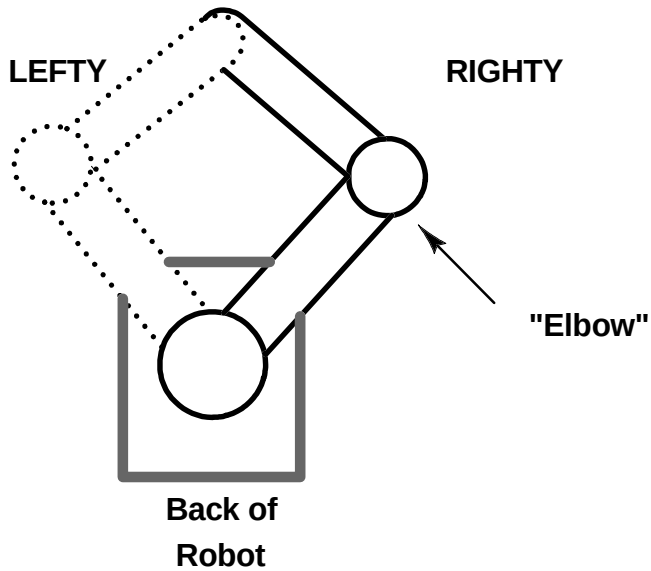


Figure 2-5
Left/Right Robot Arm Configuration Option

To display the configuration options, choose **Configuration**.

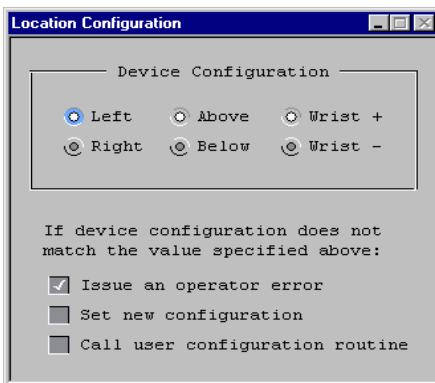


Figure 2-6
Setting Robot Arm Configuration

In some cases, the robot arm configuration (when the location is taught) may not be the optimal configuration¹ to use when the robot moves back to that location from another location. (Arm configuration is recorded when either the **Here** or **Teach** button is chosen.)

If the arm configuration at a given point is critical (for example, to avoid running into obstructions in the workcell or when high precision is required), select ☒ **Issue an operator error** on the Location Configuration menu page (Figure 2-6). Then when the robot begins a move, it will first check the arm orientation. If the orientation is not the one specified, it will halt and warn the operator.

¹ The optimal configuration is the one that requires the least amount of joint movement. In most cases, the optimal configuration is achieved by not changing arm configuration during a move.

Device Configuration radio buttons indicate the desired configuration to apply to the **Set new configuration** selection. Select ☒ **Set new configuration** to force the robot into the desired configuration on the move to this location. This option should be selected only when you know that the robot can safely change arm configuration when moving to this location from any other possible location.

Select ☒ **Call user configuration routine** to call a user-written V⁺ routine to determine the proper response. The user routine is **rn.auto.config** in the file **ROBUSR.V2**.

If none of the options are selected, the arm will move to the location using whatever configuration the arm is in at that time unless the location cannot be reached in the current configuration. In this case, execution will stop and the operator will be alerted.

Using the Manual Control Pendant (MCP) to Teach Locations

To define a location using the MCP, open the location record and choose **Teach**. The MCP display has four soft buttons, NEXT, BACK, IO, and EXIT. Select a motion mode using the MAN/HALT key on the MCP, and move the robot to the desired location (see the *Manual Control Pendant User's Guide* for details).

When the robot is at the correct location and the gripper has the proper orientation, press the REC/DONE key on the MCP. The location will be stored (but not displayed in the location record). Press the NEXT or BACK soft key to select the approach or depart move (the location name and whether the location, depart, or approach is being edited is shown on the top line of the MCP). Move the robot to the desired height and press the REC/DONE key. The height will be stored (but not displayed in the location record). Once all moves have been defined, press the EXIT soft key to display the values and exit the teach mode.

Once a location has been defined, it can be edited in teach mode. Open the location record and choose **Teach**, but leave the MCP in comp mode (do not press the MAN/HALT key). The NEXT, BACK, and EXIT soft keys as well as the REC/DONE key will behave the same as when initially teaching the location, but the MCP speed bars will allow you to move to the location or depart/approach positions. The "+" side moves toward the selected position, the "-" side, toward the previous position.

2.3 The MotionWare Databases

This section describes the various databases that MotionWare uses to store information it needs to perform robot operations. If you have the vision option, there also will be a vision database (described in the *VisionWare User's Guide*).

Location

The main purpose of MotionWare is to allow you to acquire or place parts at different locations.¹ A location database contains all the locations needed by the robot to complete an operation (or the portion of an operation designated for the robot). Location databases are discussed in Chapter 5.

¹ The most common action a robot takes at a location is to acquire or place a part. However, these are not the only actions that can be taken. MotionWare provides you with ways of controlling most any type of end-of-arm tooling.

Path

Paths provide a way of controlling the actual path that a robot takes as it moves between locations. However, they are optional and need not be specified where they are not needed. In a complicated workcell where parts are being picked up from multiple part feeders, and assemblies are being brought into the workcell in random orientation, the actual path of the robot is impossible to predict. Using paths, you can specify safe paths that the robot must follow when moving from one location to another.

Paths are created by specifying multiple locations and indicating whether the robot can enter or leave the path at each specified location. Path creation is described in Chapter 7.

Frame

The origin of the world reference frame is at the base of the robot (see Figure 2-2). Unless otherwise specified, the X, Y, and Z components of a location are offsets from this point. In many cases you will want to use relative reference frames for acquiring and inserting parts. For example, you might teach all the part locations on an entire assembly and then find that the actual placement of the assembly (as it is brought into the workcell) has changed, necessitating reteaching of all the locations.

Had a reference frame been created for the assembly and all part locations taught, relative to that reference frame, only the points needed to define the reference frame would need to be retaught. When a system includes the vision option, the reference frame can be dynamically updated by the vision system each time a new assembly enters the workcell.

Reference frames are commonly used for racks of part feeders that might not be put back in the same location each time they are refilled.

Reference frames are discussed in Chapter 8.

Tool

When a robot location is recorded, the location recorded is the actual center of the robot tool flange. When a gripper is attached to the tool flange, the center of the gripper is offset (and possibly rotated) from this center. To be accurate, this offset must be factored out of the taught location. A tool transformation (Tool) comprises the X, Y, Z, and rotation values required to factor out the offsets. A tool must be defined if multiple grippers, the vision option, or reference frames are used.

Tools are discussed in Chapter 8.

Conveyor

The conveyor database allows you to define and calibrate the different types of conveyors that MotionWare can use. The conveyor database is discussed in "Conveyor Database, Frame Database, and Location Database" on page 140.

Force

If your system is equipped with the optional AdeptForce hardware, the Force database stores information on the various force processes. Force sensing may also be used with a simple binary switch connected to one of the fast digital inputs.

Variable

There are many cases where sequences can use numeric arguments. The Variable database allows you to specify the type of numeric data and the source of the value for that data. The Variable database is discussed in Chapter 6.

Vision and Camera

If you have the optional vision extension to MotionWare, the Vision and Camera databases will store information about the vision process. This data is used to guide your robot and make inspections of objects in the workcell. These databases are described in the *VisionWare User's Guide* and are referred to in the vision-related chapters in this manual.

2.4 Preview: MOVE Statement

Once you have created all of the necessary locations for an assembly and specified any other information required to complete a robot operation, an AIM sequence must be created to perform the work. A sequence is a series of statements, each of which specifies one complete operation. For example, the MOVE statement defines the complete operation of moving to a location, executing a gripper action, and then optionally moving to another location and executing another gripper action. Sequence creation is described in Chapter 4.

The following is a quick overview of the basic MotionWare MOVE statement (excluding the vision arguments, which are described in Part Two of this manual):

1. If an optional tool has been defined, apply that tool transformation.
2. If the optional first location is defined:
 - a. If an optional along path is specified, move along that path to the first location. Otherwise, move directly to the first location.
 - b. Execute the specified gripper action.
3. If an optional approach path is specified, move along that path to the second location. Otherwise, move directly to the second location.
4. Execute the specified gripper action.
5. If an optional depart path is specified, depart from the assembly area along that path.

The following is an example of a MOVE statement:

```
MOVE APPROACH path1 FROM loc1 ALONG path2 TO loc2 DEPART path3  
USING TOOL
```

All information necessary to complete a MOVE statement is created in the MotionWare window and mouse environment. This sequence of statements also can be created using the point-and-click feature of MotionWare.

When first created, locations can be left undefined, i.e., they exist, but without values recorded in the record. The MotionWare walk-thru training option allows you to record values in undefined record locations during the first execution of a sequence (see "Debugging Sequences" on page 66).

MotionWare Resource Modules

3

3.1 What Is a Resource Module?	40
Loading and Selecting Resource Modules	41
Resource Module Directory	42
3.2 Managing Resource Modules	42
3.3 Resource Module Options	42
Module Resources	45
Global Databases	47
Dual-Vision Systems and AIM	47

3.1 What Is a Resource Module?

MotionWare is designed to efficiently manage information required to complete workcell automation jobs. In MotionWare there are two primary types of information required to perform workcell tasks:

Data – This includes the location values the robot will move to, the definition of paths it will move along, vision operations it might use to locate parts, and other information it will need to perform its work. Data is stored in MotionWare databases.

Logic – This tells MotionWare the order in which to perform each of the individual operations required to complete a task. This includes synchronization (such as waiting for a digital signal), starting other sequences at the required time, and other options that control the “when” of a job rather than the “what or how”. In MotionWare, logic is stored in sequences.

For example, a robot is required to move to a location, pick up a part, move to a second location, place the part, and then move to a safe location to allow other workcell activity. The locations the robot moves to and the activity associated with each location are stored as a data resource. This data resource does not indicate the order in which the locations are to be accessed: A logic resource is required to specify the order of events.

In the above example, you would need the following resources:

- A Location database to record the location values and gripper activity
- A Path database to record the points of the paths the robot can move along
- A sequence to define the order of events

If we expanded the above example to require reference frames for the locations and complicated gripper logic at the robot locations, we would need to add the following resources:

- A Frame database
- A sequence to define the gripper activity

If we further expanded the above example to use vision guidance, we would need to add a Vision database and a Camera database to our resources.

You can design as many resource modules as you need, but you can select only one resource module at a time. Only the sequences associated with the currently selected module can be edited. For example, consider a workcell that produces three different parts. The resource modules for the three different parts might look like Figure 3-1.

The modules use databases as follows:

- Module 1 uses its own databases except Tool, Variable, and Conveyor.
- Module 2 uses all global databases except Location and Path.
- Module 3 uses the Tool global database. If vision is installed and a Vision database has been added to the module, the Camera global database also is used.

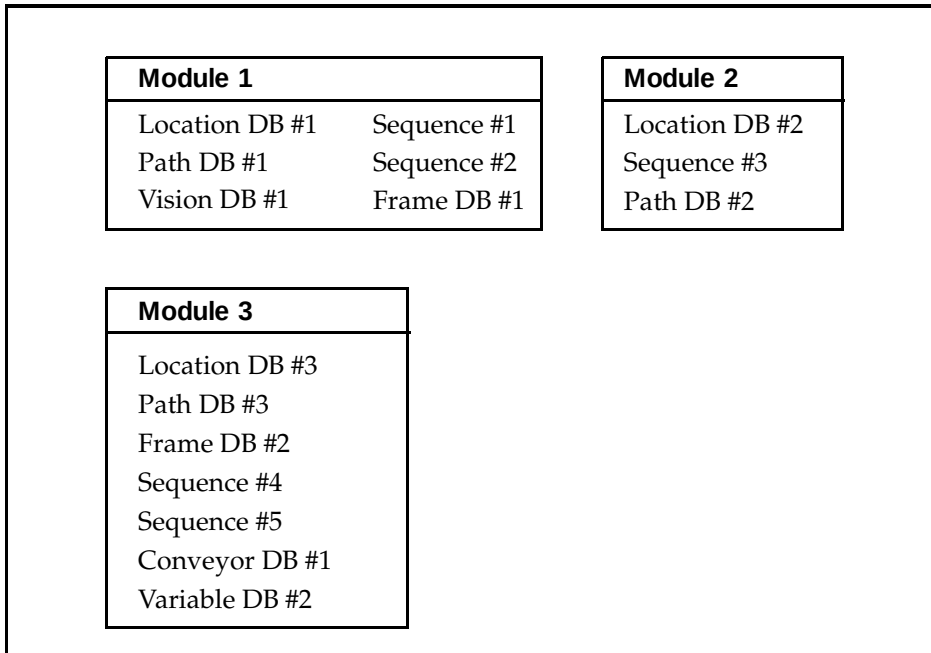


Figure 3-1
Example Resource Modules

Loading and Selecting Resource Modules

A resource module can be either unloaded or loaded. An unloaded module resides entirely on disk and uses no V⁺ memory. It can be edited with the module utility, but the data in the module databases cannot be accessed by AIM.

A loaded module is read from disk into V⁺ memory. The amount of V⁺ memory in your system limits the number of modules that can be loaded. Once loaded, the databases in the module can be accessed by the operator interface or the various runtime tasks.

Once a module is loaded, it must be “selected” or assigned to specific AIM tasks. Each task may have, at most, one module assigned to it. A module can be assigned to more than one task. Once a module is assigned to a task, any references to a database of a given type refer to the module database member of that type. If the module does not contain such a database, the corresponding global database is referenced.

A module is assigned to a runtime task by choosing the **Select** button on the operator control panel or by executing the “SELECT_TASK” control sequence statement. To edit a database within a module, you must first assign it to the operator interface by **Module ➔ Select/Edit**. When a runtime task is displayed by an operator control panel, the module assigned to that runtime task also becomes assigned to the operator interface.

The name of the module currently assigned to the operator interface is shown in the top line of the window, near the right side. If no module is assigned to the operator interface, the name ***Global*** is shown. As for a runtime task, any reference to a database accesses the components of this module. These references include the database editing pages found under the **Edit** pull-down, the sequence editor, and various set-up utilities.

If a module is assigned to an active runtime task, it cannot be unloaded or edited with the module utility.

Resource Module Directory

When you first create a module, you will be prompted to enter a directory name for storing the module resources.

The path to this directory can be absolute, such as:

`c:\data\test.data`

or

`\data\proj1\`

Or, the path can be relative to the AIM programs directory:

`data\vis_proj\`

3.2 Managing Resource Modules

Resources are created, copied, edited, loaded, and selected in the resource module utility.

To access a resource module:

Module ➔ Select/Edit

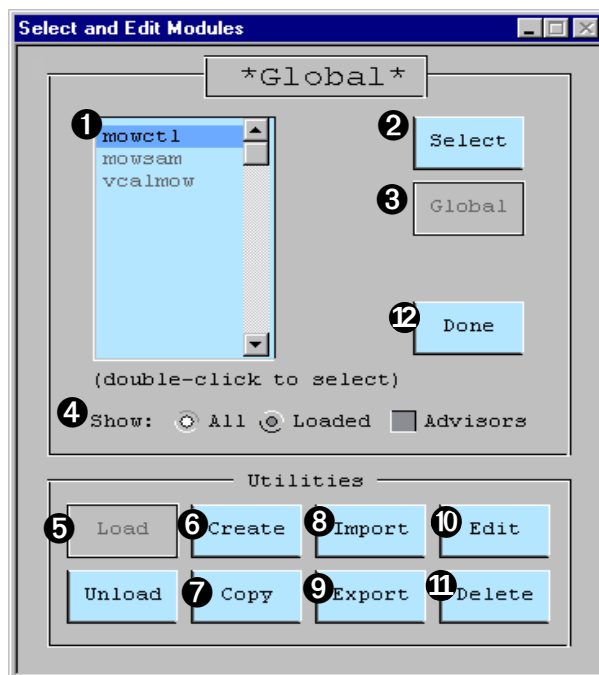


Figure 3-2
New Module Menu Page

3.3 Resource Module Options

Refer to Figure 3-2.

- ① Shows all the defined modules (if ☒ **All** is selected from item ④) and the loaded modules (if ☒ **Loaded** is selected from item ④). It also shows “All” or “Loaded” Advisor modules (if ☒ **Advisors** is selected from item ④).

Double-click a module name to select that module (and load the module if not already loaded). The currently selected module is shown on the top-level menu bar. (If no module is selected, ***Global*** is displayed.)

The currently selected module is shown in blue. Loaded modules are shown in black, and unloaded modules are shown in gray.

- ② Choose **Select** to make the highlighted module the currently selected module. A module must be selected in order to access its associated databases from the sequence editor. (If, for example, you select a module that has a Variable database in its resources, you can access the variable names in this database from the sequence editor.)

NOTE: When a module is selected, you can still access variables in the global database from the sequence editor by typing a colon ":" in front of the variable name.

- ③ Choose **Global** to deselect the current module. This allows you to access all of the global databases. (If, for example, you have a module selected that has a Path database in its resources, you cannot access the global Path database.)

NOTE: There is no global Vision or global Location database.

- ④ These radio buttons determine which module is displayed in item ①.

- ⑤ Choose **Load** to load the selected module.

Choose **Unload** to unload the selected module.

- ⑥ Choose **Create** to display a window requesting the new module name and the directory path to the module. All module components will be stored in this directory. If the directory path is left blank, the module and its components will be stored in the current directory (see "Resource Module Directory" on page 42). See the following sections (in this chapter) for details on adding components to the module.

- ⑦ Choose **Copy** to create a copy of the selected module.

- ⑧ Choose **Import** to import an entire module from a module archive file. If you want to replace an existing module, you must delete that module before using the import option. Otherwise, you must assign a new name to the module file being imported using the "Override module name:" field.

- ⑨ Choose **Export** to export a module file to disk. The entire module and all associated databases are saved to a single module archive file with a .MOD extension.

- ⑩ Choose **Edit** to add, delete, or change characteristics of module components.

- ⑪ Choose **Delete** to delete the selected module and all of its resources.

- ⑫ Choose **Done** to exit the resource module utilities.

When either a module has been first created or the **Edit** button is chosen, the following window is displayed. From this window you can add the required components. This module already has one sequence.

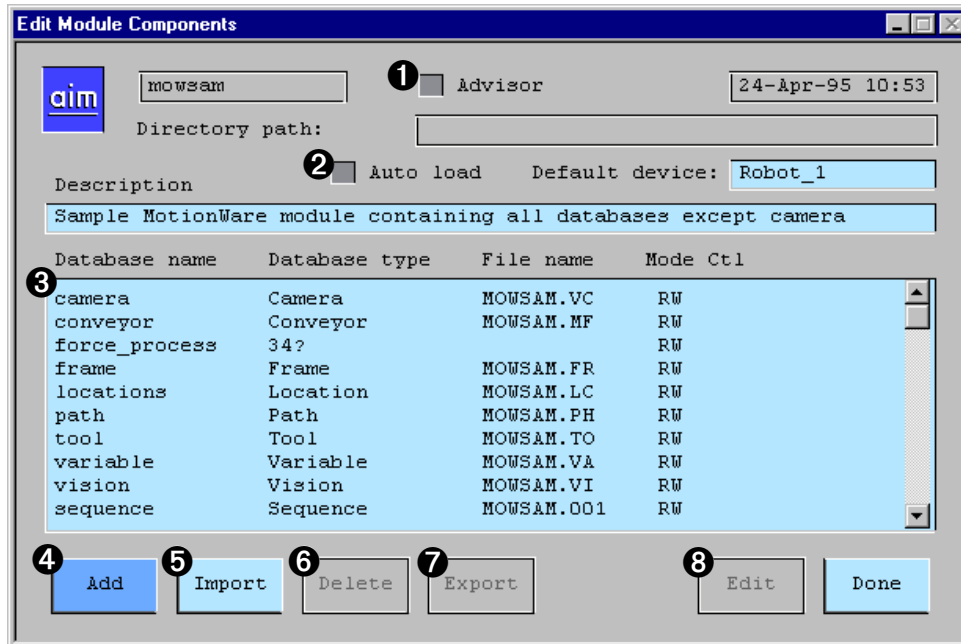


Figure 3-3
Edit Module Components Menu Page

- ❶ Select ☒ **Advisor** to mark the module as an Advisor module. This feature currently applies to the FlexFeedWare module only. Refer to the *FlexFeedWare User's & Reference Guide* for details.
- ❷ Select ☒ **Auto load** to enable the auto-load feature. Modules with this option enabled will load automatically when MotionWare is started.
- ❸ Shows all the module components and the component type, disk file name, editing mode, and control flag (when appropriate). The selected component is the current component.
- ❹ Choose to add a new component to the module (the component resource options are shown in "Module Resources" on page 45).
- ❺ Choose to import a component resource that is stored in AIM database format. (Generally, you would import resources that have previously been exported—see item ❷). You may also import a file from a different module.
- ❻ Choose to delete the selected component.
- ❼ Choose to export a component to an AIM database file.
- ❽ Choose to edit the current component characteristics.
- ❾ Choose when you are done editing components.

Module Resources

To request a new module resource, choose **Add** from the Edit Module Components window to display the following window, from which you can select module resources:

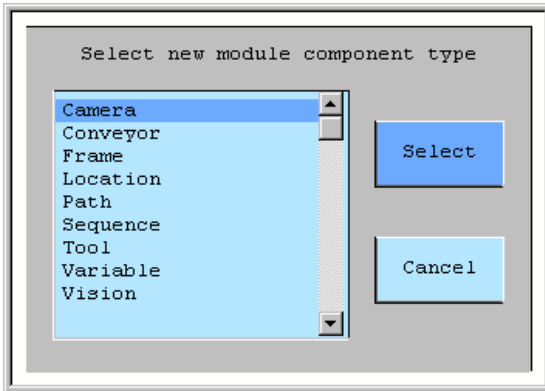


Figure 3-4
Select New Module Component Type

The resource types are:

Resource Type	Definition
Camera	Defines the camera parameters and calibration for any cameras you will use with vision processes (see Chapter 14).
Conveyor	Defines and calibrates any conveyors used with a robot (see Chapter 12).
Force	Defines the force processes used with a robot (see Chapter 11).
Frame	Defines reference frames used with a robot. Refer to Chapter 8 for more information. Also defines objects for conveyors. See Chapters 12 and 15 for more information.
Location	Defines robot locations. Refer to Chapter 5 for more information.
Path	Defines paths that the robot must use when moving between locations. Refer to Chapter 7 for more information.
Sequence	Specifies the logic for a workcell task. Refer to Chapter 4 for more information.
Tool	Defines the tool offsets for a robot. Refer to section “Tool Transformations” on page 102 for more information.
Variable	Defines variables names for use in statement arguments that can accept a numeric value. Refer to section 14.6 on page 203 for more information.
Vision	Defines the vision processes used in inspections and robot guidance. Refer to Chapter 14 for more information.

Selecting a module component brings up the following window:

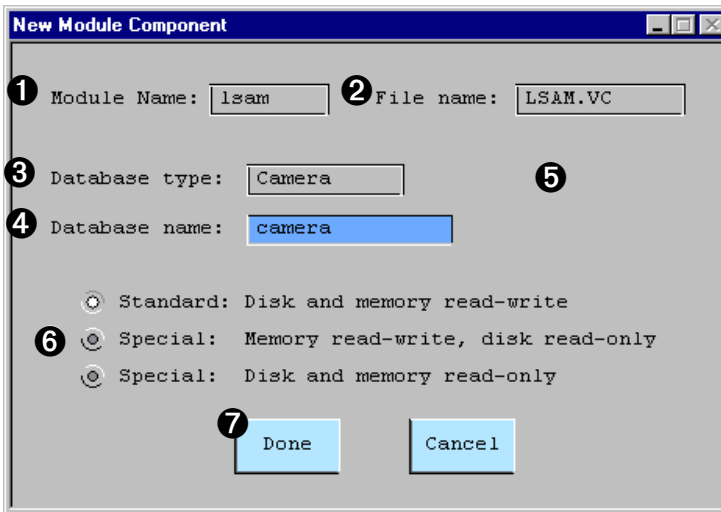


Figure 3-5
Edit Module Component

- ① Shows the name of the module to which this database will be assigned.
- ② Shows the name of the disk file into which this database will be stored.
- ③ Shows the name of the resource type for this database.
- ④ Enter a name for this new database.
- ⑤ If this is a sequence, radio buttons are displayed (in this area) that will allow you to indicate whether this should be a control sequence. Refer to page 50 for more information about control sequences.
- ⑥ Indicate the editing options for this database:
 - **Disk and memory read-write** means that the database can be edited and changes saved to disk. You must select this option to permanently add data to your database.
 - **Memory read-write, disk read-only** means that the database can be edited for the current session, but the changes will not be saved to disk.
 - **Disk and memory read-only** means that the database cannot be edited.

Once a database has been created, the read-write options can be changed by:

Module ➔ **Select/Edit** ➔ *highlight module database is assigned to* ➔ **Edit** ➔ *highlight database* ➔ **Edit** ➔ *select edit option* ➔ **Done**
- ⑦ Choose when you have selected the desired options.

Global Databases

In all cases, except Location and Vision databases, there is a global database that can be used by a sequence when its associated module does not specifically include a copy of a database. For example, Module 2 from Figure 3-1 does not include a Frame database. If a statement in the sequence associated with Module 2 specified a frame record, it would come from the global Frame database.

If a sequence needs location or vision records, the resource module must have a location or vision database among its resources. If the database is not in the resource module, you will get a **Required database not loaded** error when attempting to complete statements that require location or vision records.

Dual-Vision Systems and AIM

Adept controllers are available with a dual-vision system option. If the second vision system is installed, you will be able to select a vision system to associate with a vision database. This selection is made when you add a vision resource to a module.

If your module also includes a camera database, the vision system selected for all camera records **MUST** match the one selected for the vision database. You cannot access different vision systems from a single-vision database. You specify the vision system to access when you create the resource module component.

If you want to share a global camera database from two (or more) different vision databases (which may each use a different vision system), specify the vision system on each record of the camera database and then ensure that the camera record references a particular vision database and has a matching vision system. If you make a mistake, you will see warning signs and get linking errors.

Sequence Creation and Execution

4

4.1 Introduction	50
4.2 Sequences and Resource Modules	50
Control Sequences	50
Creating a Sequence	50
The Sequence Index.	50
4.3 Editing a Sequence.	52
Selecting a Statement.	52
Statement Components	52
Completing a Statement Argument.	53
Adding a Sequence Line.	53
Completing Optional Statement Arguments	53
Optional Clauses	53
Completing Constant Arguments	54
Adding Comments to a Sequence	54
Removing Sequence Lines.	54
Cutting, Copying, and Pasting Statement Lines	55
Copying Statement Lines Between Sequences	55
Sequence Editor Special Keys	56
Finding and Finding/Changing Statement Arguments	59
4.4 What Are the Sources for Statement Arguments?	60
Avoiding Trouble With Statement Arguments.	61
4.5 The Statement Tree	62
Statement Tree Options	63
Source Records	63
4.6 Task Control Panel	64
Task Control Panel Options	64
4.7 Debugging Sequences	66
Task Control Debug Options	67
Controlling the Robot	68
4.8 Using AIM Tasks.	68
How Do I Decide Which Tasks to Run?	69
How Do I Start a Task?.	69
4.9 Master Control Panel	70
4.10 Editing Database Records During Execution	71
Activating Single-Step Execution	72

4.1 Introduction

There are four primary steps to building a MotionWare workcell implementation:

1. Create a resource module that identifies the resources associated with a particular job. This process is described in Chapter 3.
2. Create records for all the locations, paths, and other items that are needed for the particular job. Creating these records is detailed in Chapters 4-6.
3. Create a sequence and add the statements that actually use these records to generate actions in the workcell. Creating a sequence is described in the next several sections.
4. Execute the sequence of statements. The control panels that are used to initiate, control, and debug sequences are described in sections 4.6, 4.7, 4.9, and 4.10.

4.2 Sequences and Resource Modules

All sequences must belong to a resource module. The resource module determines which databases are available to a sequence. Sequences are available for editing only if they belong to the currently selected resource module. See Chapter 3 for details on creating and managing resource modules.

Control Sequences

One of the options available when adding a sequence to a module is to make it a control sequence. Control sequences manage other sequences and perform tasks such as loading, selecting, starting, and stopping operation sequences in the workcell. Control sequences lock out all operator input while they are being executed; they cannot move the robot or access vision data. Consequently, you should set up control sequences so that they execute quickly. See “Sequence Control Statements” on page 185 for additional details.

Creating a Sequence

Sequences are created from the module utilities (see Chapter 3) or the sequence editor (see below). A sequence created from the module utilities is assigned to the module you are currently editing. A sequence created from the sequence editor is assigned to the currently selected module.

When you create a new sequence, the Edit Module Component Window shown in Figure 3-5 is displayed.

The Sequence Index

Sequences can be created, deleted, or selected for editing from the Sequence Index. New sequences created from the Sequence Index will be assigned to the currently selected module.

To display the Sequence Index from the main menu:

Edit ➡ Sequence

To display the Sequence Index from the Sequence Editor menu:

Go ➡ Sequence Index

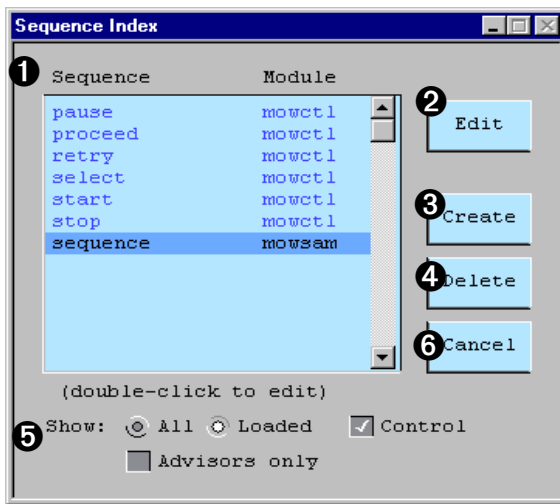


Figure 4-1
Sequence Index

- ❶ Shows all the currently defined sequences and the module they are assigned to. A sequence is automatically loaded when its assigned module is loaded. Double-click a sequence to open the sequence editor. Loaded sequences are shown in black, unloaded sequences are gray, and the sequences in the selected module are blue.
- ❷ Choose to open the Sequence Editor and begin editing the selected sequence.
- ❸ Choose to create a new sequence. The window shown in Figure 4-1 is displayed.
- ❹ Choose to delete the selected sequence.
- ❺ Indicate which types of sequences should be displayed.

NOTE: Use of ☒ **Advisors only** is not supported in AIM 3.1.
- ❻ Choose to exit the Sequence Index window.

4.3 Editing a Sequence

Before a sequence can be edited it must be created and loaded.

When a new sequence is first opened in the Sequence Editor it will appear as:

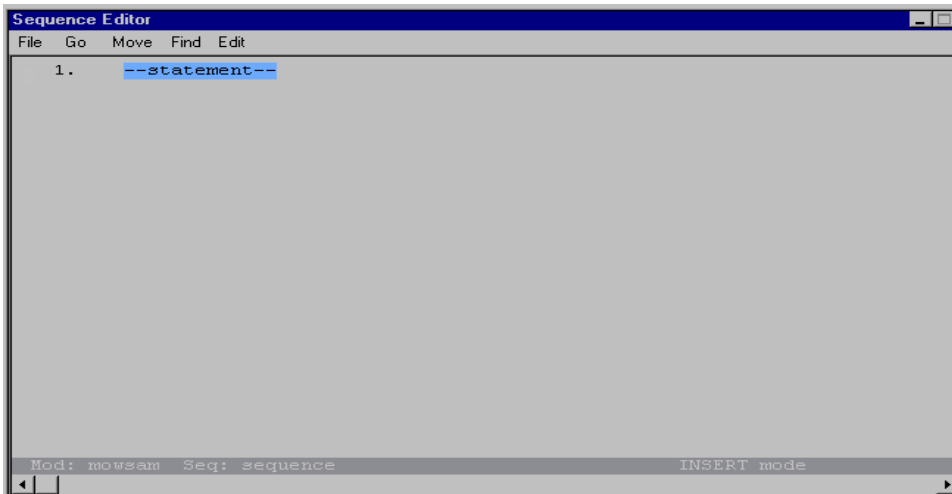


Figure 4-2
Sequence Editor

The bottom of the Sequence Editor shows the module that is associated with this sequence, the name of the sequence, and the editing mode.

NOTE: The Sequence Editor screen displays up to 41 text lines. If your sequence has more than 41 lines, use the scroll bars to view the additional lines.

Selecting a Statement

Double click `--statement--` (or perform **Go ➔ Display Field Index**), to display a list of available statements. Select the desired statement and choose **Retrieve** (or double click the desired statement).

For example, if the statement `MOVE` had been selected, the sequence editor would look like the following:



Statement Components

In the above example:

`MOVE` is the statement name.

`TO --location--` is a clause.

`--location--` is an argument.

Completing a Statement Argument

Double-click the statement argument to display a list of available records that can be used to fulfill the statement argument. Select the desired record and either choose **Retrieve** or double-click the argument to use that record as an argument to the statement.

For example, if **place.loc** had been selected, the sequence editor would look like:

```
1.  MOVE TO place.loc
```

Adding a Sequence Line

Sequence lines can be added either to the bottom of the sequence or above the current sequence line. (The current sequence line is the one with any argument or statement name selected.)

To append a sequence line to the end of the sequence, press the NEW key (F2) or:

Edit ➡ Append line

To add a sequence line above the current statement line, press CTRL+INS or SHIFT +INS or:

Edit ➡ Insert line

If **Insert line** had been chosen, the Sequence Editor would look like:

```
1.  --statement--
2.  MOVE TO place.loc
```

Completing Optional Statement Arguments

Most statements will have both optional and required arguments. Optional arguments are not displayed until the statement line is expanded.

To expand a statement, select the statement line and:

Edit ➡ Expand (or double-click the statement name)

If a WAIT_FOR statement had been added to the above sequence and then expanded, the sequence would look like the following:

```
1.  WAIT_FOR --input-- {OUTPUT --output-- {PULSE --constant--}}
2.  MOVE TO place.loc
```

Optional Clauses

Optional clauses are surrounded by brackets ({}). In the above example, PULSE is a nested clause (its brackets are enclosed within another set of brackets). In this case, the OUTPUT clause must be completed before the value in the PULSE clause will have any effect.

Completing Constant Arguments

In many cases, the argument requires a value or variable name rather than a record name. To complete this argument, select the argument and enter a value. Or double click the argument to display a list of keywords or variable records. See “What Are the Sources for Statement Arguments?” on page 60 for details on the sources for statement arguments. Double click the argument to display a list of keywords.

Adding Comments to a Sequence

Descriptive comment lines can be added to a sequence by first typing a semicolon in the `--statement--` field followed by a TAB. You can then enter up to 80 characters of descriptive text.

```
1.; ABSTRACT - Example Robot sequence to pickup one part from FF250_1
2.;           and place to a static drop off point.
3.;
4.   FF_STATISTICS FEEDER 1 ROBOT yes RESET yes
5.   WHILE -1
6.; Pull one part from the 'q' and place results in variable frame type.
7.   GET_FRAME ffl_var_frame 1: ffl_part_1 WAIT yes WAIT_PATH
   yes OUTPUT conv_bitmask
8.; Pick one part from the feeder and drop it off in static drop off.
9.   MOVE FROM ffl_pickup TO ffl_drop
10.  FF_STATISTICS FEEDER 1 ROBOT yes
11.  END
```

A well-documented sequence will be extremely helpful in debugging and updating your sequences.

Removing Sequence Lines

Sequence lines can be permanently removed from a sequence by selecting the line to be removed and then pressing CTRL+DELETE or:

Edit ➡ Delete line

Sequence lines can be made temporarily nonexecutable by disabling them. When a sequence line has been disabled, an asterisk will appear in front of the line, and the line will be ignored when the sequence is run.

To disable a sequence line, select the line and then:

Edit ➡ Disable (or double-click the number next to the statement)

After the sequence line is disabled, the sequence looks like the following:

```
1.*  WAIT_FOR --input-- {OUTPUT --output-- {PULSE --constant--}}
2.   MOVE TO place.loc
```

To remove the disable marker and make the statement line executable, select the disabled line and:

Edit ➡ Disable (or double-click the number next to the statement)

Cutting, Copying, and Pasting Statement Lines

To cut a sequence line and move it to another location in the sequence, select the line and press the CUT key (SHIFT+F9), or:

Edit ➡ Cut Line

The selected line will be removed from the sequence, and the number 1 will be displayed in the lower right corner of the Sequence Editor.

To paste the line to its new location, select the line below the paste location and press the PASTE key (F10), or:

Edit ➡ Paste Line

To copy a sequence line to another location in the sequence, select the line to be copied and press the COPY key (F9), or:

Edit ➡ Copy Line

The selected line will be copied, and the number 1 will be displayed in the lower right corner of the Sequence Editor.

To paste the copied line, select the line below the paste location and press the PASTE key (F10), or:

Edit ➡ Paste Line

More than one line can be cut or copied at a time. Each time a line is cut or copied, it is placed in a copy buffer. The number of lines copied is shown in the lower right corner of the Sequence Editor.

When

Edit ➡ Paste Line

is performed, the least recently copied or cut line is pasted above the current statement line.

When

Edit ➡ Paste All

is performed, all the lines in the copy buffer are pasted above the current line.

Copying Statement Lines Between Sequences

Lines also can be pasted to different sequences. If you exit the sequence editor while there are lines in the copy buffer, they still will be available for pasting the next time the editor is accessed. The lines will be available regardless of which sequence is selected for editing.

Sequence Editor Special Keys

The following special key assignments apply when the sequence editor is open:

Special Key	Purpose/Use
NEW (F2)	Adds a new line to the end of the sequence.
SAVE (SHIFT + F2)	Displays the Save All Modified Databases window.
GOTO (F3)	Displays the list of statements if <code>-- statement --</code> is selected. Displays the list of keywords or variables if a statement argument that requires a keyword or variable is selected. Displays a list of record names if the argument is selected but not defined. Goes to the specified record if the argument has been defined.
RETRIEVE (SHIFT + F3)	Returns you to the sequence editor and completes the statement argument with the record you were editing (if you have opened a database editing page by double-clicking a record argument or by pressing F3).
EXIT (F4)	Closes the sequence editor.
DISPLAY (F5)	Displays a list of keywords, variables, or record names if a statement argument is selected. Displays a list of available statements if <code>-- statement --</code> or a statement name is selected.
UNDO (F6)	Performs an undo on any editing changes made to the <i>current</i> field.
FIND (F7)	Opens the find window.
CHANGE (SHIFT + F7)	Opens the change window.
REPEAT (F8)	Repeats the most recent Find or Change operation.

Special Key	Purpose/Use
COPY (F9)	Places a copy of the current statement in the copy buffer.
CUT (SHIFT + F9)	Removes the current statement and places it in the copy buffer.
PASTE (F10)	Pastes the last line in the copy buffer above the current statement.
PASTE ALL (SHIFT+F10)	Pastes all lines in the copy buffer above the current statement.
EDIT (F11)	Expands/contracts the current statement.
HOME	Goes to the first statement line.
END	Goes to the last statement line.
PG UP	Goes to the top line displayed in the sequence editor.
PG DN	Goes to the last line displayed in the sequence editor.
TAB	Moves to the next editable item in the statement.
SHIFT + TAB	Moves to the previous editable item in statement.
CTRL + DELETE	Deletes the current statement line.
ESC	Toggles the disabled statement marker.
DELETE	Deletes the character under the typing cursor.
CTRL + INS	(On the numeric keypad) inserts a statement line above the current line.
←	Moves the typing cursor left.
→	Moves the typing cursor right.

Special Key	Purpose/Use
↑	Moves to the previous statement line.
↓	Moves to the next statement line.
Double-clicking	A statement name: expands/contracts the statement. A statement number: toggles the disabled marker. An undefined argument: displays the list. A defined keyword or variable: displays the list. A defined record name: opens record editing. A - - statement - -: displays the list of statements.

Finding and Finding/Changing Statement Arguments

The Find and Change options are useful for editing long sequences.

To find a statement or argument name, press the FIND key (F7) or:

Find ➡ Find setup

and the first window shown in Figure 4-3 is displayed.

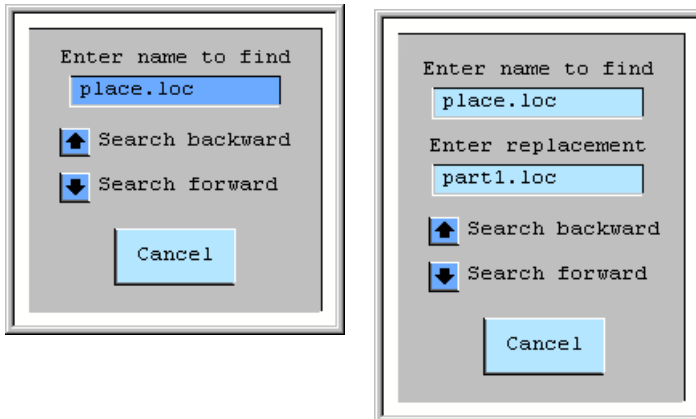


Figure 4-3
Find and Change Windows

Fill in the **Enter name to find** data box with the text you want to find. Click the up arrow to search from the current location to the top of the sequence. Click the down arrow to search from the current location to the bottom of the sequence.

To repeat the last find operation, press the REPEAT key (F8) or:

Find ➡ Repeat Find

You also can search for statement or argument names and replace them with other statement or argument names using the CHANGE key (SHIFT + F7) or:

Find ➡ Change Setup

and the second window shown in Figure 4-3 is displayed.

Fill in the **Enter name to find** data box with the text you want to find. Fill in the **Enter replacement** data box with the text to replace the found text with. Click the up arrow to search from the current location to the top of the sequence. Click the down arrow to search from the current location to the bottom of the sequence.

To repeat the last find operation, press the REPEAT key (F8) or:

Find ➡ Repeat Change

4.4 What Are the Sources for Statement Arguments?

When you first bring a new statement into the sequence editor, each statement argument has a place holder identified by pairs of dashes (-- argument --). These place holders let you know the legal sources for arguments to replace the place holder. The source may be a database, special keyword, or value that you enter directly. The following table describes the legal sources for the various placeholders.

NOTE: The argument field width is 18 characters.

Table 4-1
Statement Argument Sources

Place Holder	Source
--component--	Keyword from the list of Cartesian components (X, Y, Z).
--constant--	Record from the variable database specifying an integer constant. A integer value entered from the keyboard.
--conveyor--	Record from the conveyor database.
--database--	The name of a standard AIM database type.
--device--	Keyword from the list of devices.
--event--	Keyword from list of pause-after keywords.
--force--	Record from the force database.
--frame--	Record from the frame database.
--I/O_mode--	The I/O mode used when opening an I/O server device. May have the values "read", "write", or "read_write".
--input--	Record from the variable database specifying an Input signal. Valid signal number entered from the keyboard (can be preceded by an optional "@").
--location--	Record from the location database.
--module--	Module name from list of defined modules. Module name entered from the keyboard.
--mode--	Keyword from a list of V ⁺ motion-nulling modes. Values may be coarse, fine, or none.
--opr--	Keyword from the list of arithmetic and relational operators. Legal operator entered from the keyboard.
--output--	Record from the variable database specifying an Output signal. Valid output signal number entered from the keyboard (can be preceded by an optional "@").

Table 4-1
Statement Argument Sources (Continued)

Place Holder	Source
--o_variable--	Record from the variable database that specifies an Output signal, V ⁺ variable, or ai.ctll[] value. Valid output signal number entered from the keyboard (must be preceded by "@").
--path--	Record from the path database.
--protocol--	The name of an I/O protocol used with the I/O server statements. The standard Adept-supplied protocols are named "simple_serial" and "var_len_file".
--response--	Keyword from list of operator responses to paused sequence.
--s_opr--	A binary string operator. May have the values "+", "LEFT_OF", "RIGHT_OF", "LEFT_OF_ANY", "RIGHT_OF_ANY", "LEFT", "RIGHT", "REMOVE", "REMOVE_NOT".
--s_uopr--	A unary string operator. May have the values "UPPER" or "LOWER".
--segment--	Keyword from the list of robot move segments (approach, depart, location).
--sequence--	Sequence from the list of sequences in the currently selected module. Sequence name entered from the keyboard.
--string--	Character data entered from the keyboard.
--string_var--	Quoted string (up to 15 characters) or the name of a string-valued AIM variable.
--task--	Keyword from list of task keywords.
--tool--	Record from the tool database.
--uopr--	Keyword from the list of unary operators. Valid unary operator entered from the keyboard.
--variable--	Record from the variable database. Numeric value (integer or real) entered from the keyboard. Digital signal (preceded by "@").
--vision--	Record from the vision database.
--yes/no--	Keyword from "yes/no" keywords, default is "no" if omitted.

Avoiding Trouble With Statement Arguments

When you enter a record name or value into a statement, the sequence editor checks the argument type and range. It then issues a warning if there is a problem. However, once

the argument has been accepted, there are several possible changes you can make that will result in error messages that will show up at run time or when a sequence is linked.

Some of the changes you might make that create errors are:

- Changing the type of a variable record
- Deleting records used in a sequence
- Deleting or unloading databases or sequences that are referenced by a statement
- Removing digital I/O hardware that changes the valid range of I/O signal numbers
- If you want to explicitly enter a digital signal number in a `--variable--` field, it must be preceded by the `"@"`. Otherwise, it will be interpreted as a numeric value. For example, `"IF 1001 THEN"` is always true because 1001 is interpreted as a nonzero value. If you really want to refer to a digital signal, use `"IF @1001 THEN"`.

4.5 The Statement Tree

The statement tree allows you to see the interconnection of various records that an executing statement will access. As you build MotionWare records you will notice that each record has an icon associated with it. The statement tree makes use of these icons to show the relationships among the records used by a statement.

The statement tree is activated by:

Show ➡ Statement Tree

This menu option can be chosen at any time. The most recently selected sequence will be shown on the statement-tree menu page. When the statement tree is first displayed, a yellow box surrounds the current record.

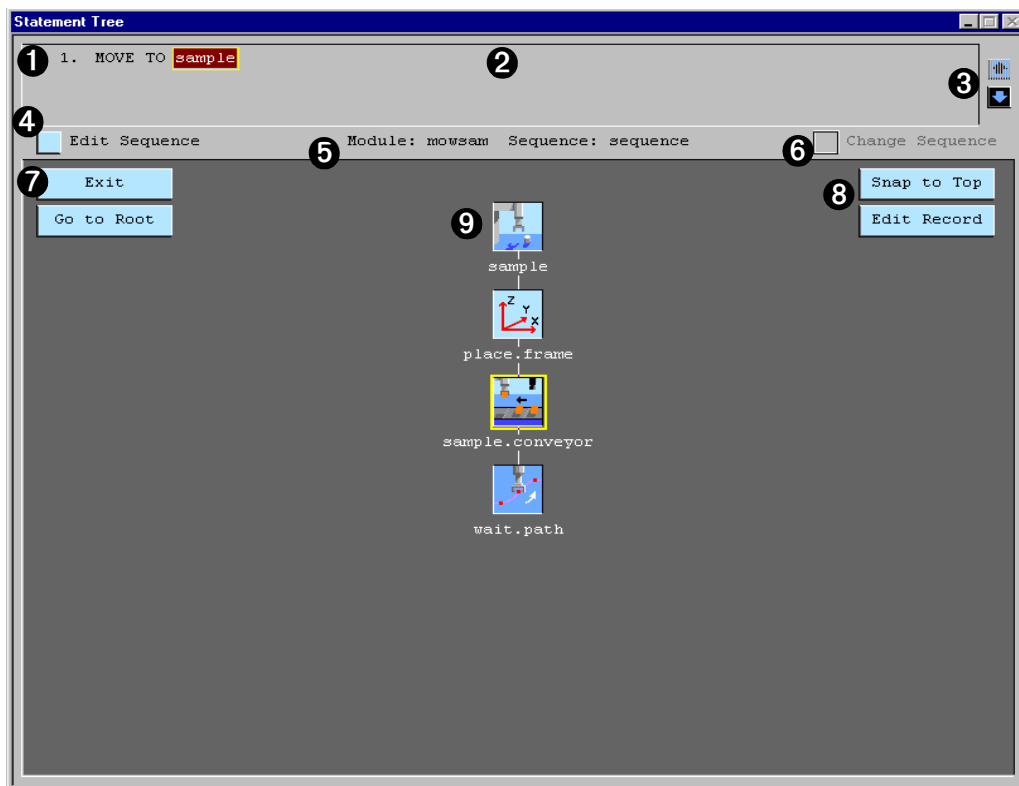


Figure 4-4
Sample Statement Tree

Statement Tree Options

- ❶ Shows the currently selected statement and its step number in the sequence.
- ❷ The selected argument indicates which argument tree is currently displayed. To display the tree for a different argument click the desired argument.
- ❸ Scroll through the statements in the sequence.
- ❹ Choose to edit the sequence.
- ❺ Shows the name of the sequence being displayed and the associated resource module.
- ❻ Choose this button to display the record tree for other sequences.
- ❼ Choose **Exit** to exit the statement tree display.

Choose **Go to Root** to display the top argument record. (This option is meaningful only if there are more source records than are displayed at one time. Arrow buttons will indicate that there are records above or below the ones being displayed.)

- ❽ Choose **Snap to Top** to move the currently selected record to the top of the display.
- Choose **Edit Record** to open the menu editing page for the selected record. You can also edit a record by double-clicking the record icon.

- ❾ The first record shown is the actual argument to the statement. Records below the first record are source records, which are records specified in the record for the next higher record in the argument tree.

If a question mark appears below the icon, it means one of the following:

- The statement requires another record, which was not specified.
- A specified record name was not created.
- A specified item was not loaded on startup.

Double-click any icon in the argument tree to go to the editing page for that record.

Source Records

Records are used either directly or indirectly by an executing sequence. The records used directly are specified as statement arguments. Records used indirectly are specified in a data box as a record referenced by a statement. These records are referred to as source records.

For example, a location record might have an associated reference frame. The frame record is never entered directly into a statement, but its data is accessed by a sequence based on the frame record that is directly referenced by the statement.

The argument tree shows all the directly and indirectly referenced records in an argument. In Figure 4-4, the **place.frame** record is a source record for the **sample** record.

4.6 Task Control Panel

Once a sequence has been created, it can be selected for execution. Its execution status is monitored from the Task Control Panel.

To display the Task Control Panel:

Execute ➡ Task Control Panel

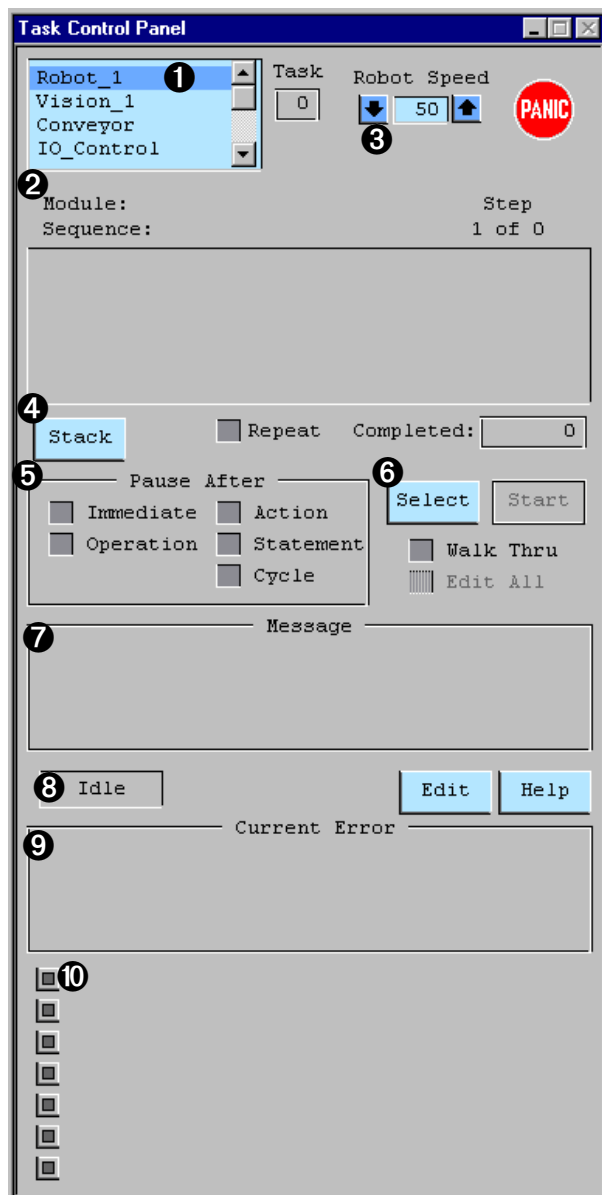


Figure 4-5
Task Control Panel

Task Control Panel Options

- ❶ This list shows the tasks that can be selected to run. See “How Do I Decide Which Tasks to Run?” on page 69 for details on selecting tasks and associated sequences. Task indicates the V+ task number that the sequence will run in.¹

NOTE: All other items on the task control panel apply only to the selected task. To view the status of another task, you must select that task.

- ② This area shows the assigned sequence, associated module, and currently executing step.
- ③ Use the arrow keys to increase or decrease robot speed or enter a value in the data box. Speed should be set low (below 25) when a sequence is first run, and it should be set above 100 very cautiously. See “Speed” on page 31 for a discussion of robot speed.
- ④ Choose **Stack** to see the current status of CALL statements executed in the current task.

Select ☒ **Repeat** to cause the currently selected sequence to loop repeatedly (the repeat option must be set for each sequence; you can have some sequences that repeat and some that do not). The Completed data box shows the number of cycles that have been completed.

- ⑤ The Pause-After options refer to places that a sequence will pause while performing debug operations. See the next section for details.
- ⑥ Choose **Select** and a list of available sequences is displayed.

Choose **Start** to either begin execution of the sequence shown in item ② or start a server task (see “How Do I Decide Which Tasks to Run?” on page 69). Server tasks do not have associated sequences.

☒ **Walk Thru** allows you to debug sequences. See “Debugging Sequences” on page 66.

- ⑦ Displays messages that are generated during the debug modes (see “Debugging Sequences” on page 66). Clicking the Message panel displays the error database help page for the current message, if any. Click this area to display help for the current message.
- ⑧ Shows the status of the currently selected sequence. Clicking this area will open the Master Control Panel (see “Master Control Panel” on page 70).

Choose **Edit** to edit the database associated with the current error, if any.

Choose **Help** to display help information associated with the current error, if any.
- ⑨ Shows the last error that was generated by MotionWare. Clicking the Current Error panel displays the error database help page for the current error, if any. Click this area to display help for the current error.
- ⑩ These buttons are used during debug operations and are described in “Debugging Sequences” on page 66. Click the button label to display help for a button.

¹ As delivered, MotionWare can use up to two robots or motion devices. MotionWare can be customized to use an arbitrary number of robots or motion devices.

4.7 Debugging Sequences

Sequences may be run in debug mode from the Task Control Panel. There are two primary debug modes: Walk-Thru training and Pause After. Walk-Thru training is primarily used to teach locations; Pause-After options are used to debug a completed sequence. Both options work in a similar fashion.

When Walk-Thru training is enabled, AIM performs additional error checking and generates trace messages that describe the actions being taken. This is useful in diagnosing problems since it may report error conditions that are normally ignored. Walk-Thru training mode is not as efficient as normal mode, so it should be disabled for non-debugging situations when cycle time is important.

Walk-Thru training is enabled separately for each task. For example, you can enable it for the robot task to diagnose a robot problem while it is disabled for vision. Walk-Thru training may also be enabled for a server task to diagnose problems related to the server operation.

When a debug mode is selected, the Task Control Panel data will appear similar to the following:

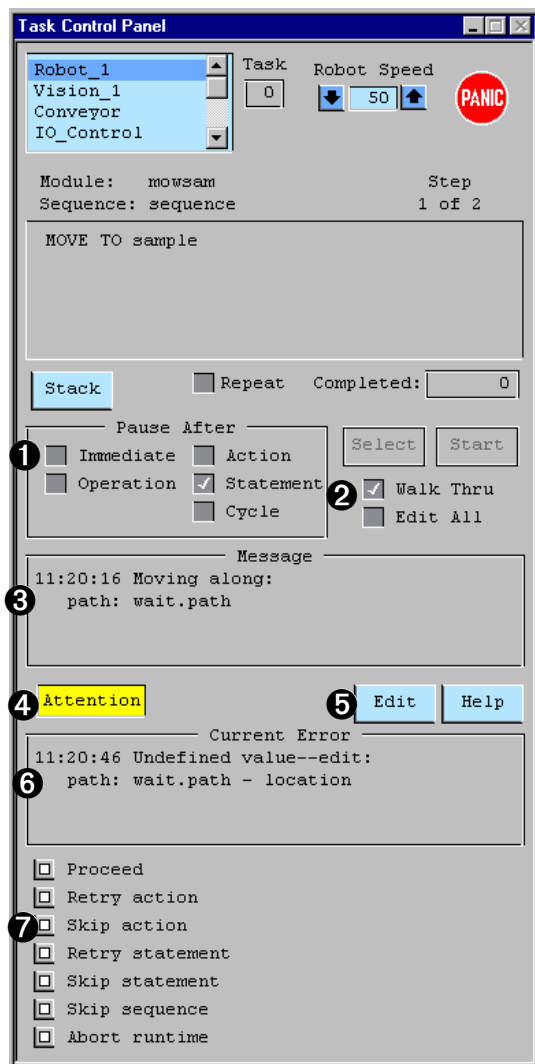


Figure 4-6
Task Control Panel With Debug Options

Task Control Debug Options

- ❶ Pause After enables you to determine when execution pauses.

The following table shows the status of the currently executing sequence:

Selection	Causes execution to pause after
Immediate	This option is selected
Operation	Each individual robot motion or vision operation
Action	Each group of robot motions
Statement	The currently executing statement completes
Cycle	The currently executing cycle of the sequence completes (cycle refers to cycles of the main sequence and not to cycles of any sequences started with a CALL statement)

A Pause-After option can be selected at any time prior to or during execution of a sequence; more than one pause point can be selected.

Pauses after Operation and Action depend on the design of each statement and, thus, do not have a rigid definition like pauses after Statement and Cycle.

- ❷ When ☒ **Walk Thru** is selected, the sequence will pause before each undefined location field referenced by any statements in the sequence.

When ☒ **Edit All** is selected, the sequence will pause before every location field referenced by any statements in the sequence.

☒ **Walk Thru** must be selected before ☒ **Edit All** can be selected.

- ❸ This area displays information on the current task activity.
- ❹ This area shows the execution status of the task. The most common states are idle, running, and attention. Attention indicates that the sequence has paused and an operator response is required.
- ❺ When a sequence has paused because of a debug operation or error, choosing this button will open the record currently being evaluated in the sequence. You will be able to edit most (but not all) of the fields in the record. There are certain fields which must have legal values before the sequence can start, and they cannot be changed during sequence execution. If you attempt to edit one of these fields, an error message is displayed advising you that the field cannot be edited.

Choose **Help** to display help information associated with the current error, if any.

- ❻ Displays the last error generated by MotionWare. This might be an execution error or a message indicating why the sequence stopped during a debug operation.

- 7 After a sequence has been paused, choose:

Option	Result
Proceed	Resumes execution.
Retry Action	Re-executes the action completed immediately before execution was paused.
Skip Action	Skips the next action in the statement.
Retry Statement	Retries the entire statement being executed when the sequence paused.
Skip Statement	Skips the statement being executed when the sequence paused.
Skip Sequence	Skips the currently executing cycle of the sequence (aborts execution at the last cycle).
Abort Runtime	Aborts runtime execution completely.

NOTE: Only the appropriate options are displayed after a pause. Clicking the label of the option button displays Help information on that option.

Controlling the Robot

Use the Speed data box to control the robot speed (enter a value directly or use the arrows to raise or lower speed). Speed should be set low (below 25) when a sequence is first run. It should be set above 100 very cautiously. See “Speed” on page 31 for a discussion of robot speed. Speed can be changed while a sequence is executing.



WARNING: Choosing the PANIC button immediately pauses an executing sequence and interrupts all tasks. The button has the same effect from both this screen and the Master Control Panel. Pressing the PAUSE key on the keyboard also stops execution.

Deselecting the ☒ **Repeat** option will stop sequence execution at the end of the current cycle.

4.8 Using AIM Tasks

AIM makes use of the V⁺ multitasking capability. It allows multiple programs to run concurrently to help your tasks run in the most efficient manner. AIM uses V⁺ tasks in four primary ways:

V+ Tasks	Use
Front-end Tasks	These tasks control the AIM environment; opening and closing windows, responding to menu choices, and managing the databases and other operations required to run AIM. These tasks run automatically without any intervention on your part.
User Tasks	These tasks run sequences, and they are further divided into tasks that control a robot and tasks that do not control a robot.
Server Tasks	These tasks perform work required if certain Adept options are employed by your user tasks. The conveyor and vision tasks are server tasks.
Background Tasks	These tasks run continuously and can perform standard operator error I/O. They never stop until AIM is shut down, but they may pause if an error occurs. These tasks must not execute sequences since they are not interlocked with the linker or with database changes.

How Do I Decide Which Tasks to Run?

The tasks you select to run are determined by which workcell activities are controlled by MotionWare. At a minimum, you will have one robot task controlling a robot or motion device. If you have a second robot or motion device, you will need a second robot task.

If user tasks execute a sequence from a resource module that contains a vision database, you must start the vision server task. Similarly, if you have statements that include records that reference a conveyor record, you must start the conveyor-server task.

Finally, you may have sequences that control I/O or other peripherals that are not associated with a robot, but must execute concurrently with the robot tasks. In such cases you must start the I/O Control task.

How Do I Start a Task?

Tasks are individually started and controlled. If you are using a server task, start it first. I/O Control tasks should be started next (if used), and the robot tasks should be started last.

To start a task:

1. Select the task you want to start (item ② in Figure 4-5).
2. Choose **Select** to display a list of available sequences.
3. Select the desired sequence.
4. Set the robot speed if you are running a robot task.
5. Select ☒ **Repeat** if you want to execute multiple cycles of the sequence.
6. Choose **Start** to begin execution of the sequence.
7. Repeat the procedure for any other tasks you want to start.

4.9 Master Control Panel

The Master Control Panel controls sequence execution. Each of the large buttons on the Master Control Panel will run a control sequence. You can edit these control sequences to perform whatever actions you desire.

To display the Master Control Panel:

Execute ➔ Master Control Panel

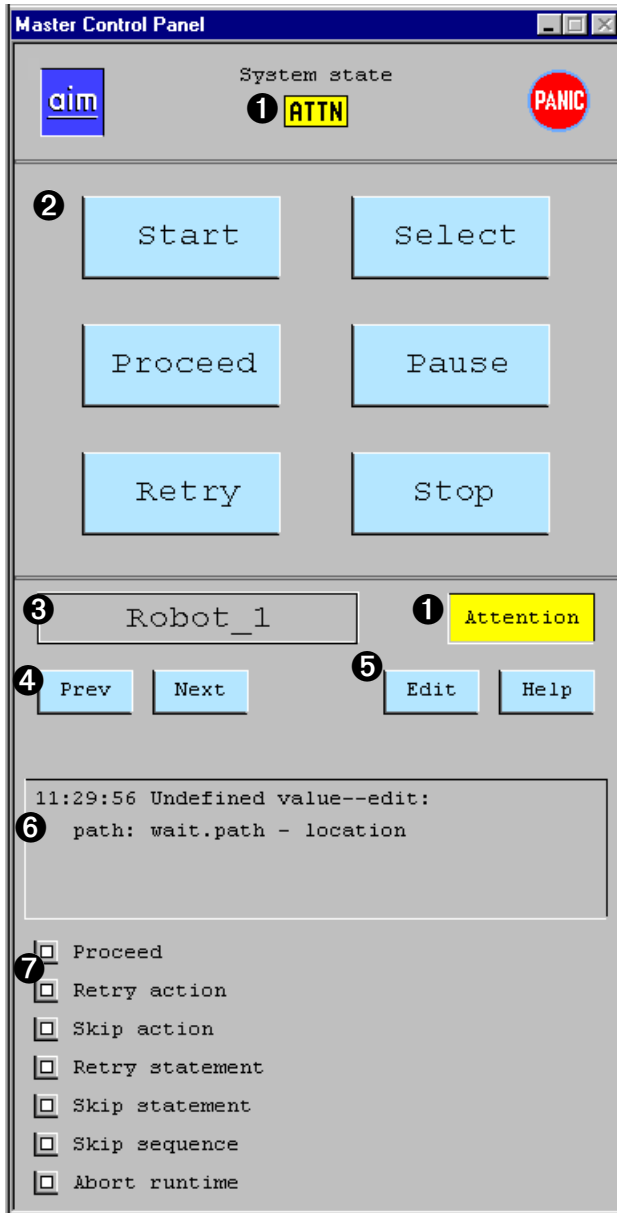


Figure 4-7
Master Control Panel

- ❶ This area shows the current state of the entire AIM system such as idle, attention, and running.
- ❷ These buttons will execute the correspondingly named control sequence. You can create and edit these control sequences to perform whatever actions are desired. For example, choosing **Start** will execute the sequence "start". This sequence could be edited either to start a sequence that has already been selected and loaded or to load, select, and start a sequence.

To use these buttons, there must be a loaded module named **mowctl** and a sequence in this module with the same name as the button. When a button is chosen, it will attempt to execute a sequence with the same name as the button. It expects the sequence to be in the **mowctl** module.¹ See "Sequence Control Statements" on page 185 for details on statements that can be used in control sequences.
- ❸ This area shows which runtime task is reporting an error (displayed in item ❹). Clicking this area will display the Task Control Panel.
- ❹ These buttons scroll through the various tasks that have paused and need attention.
- ❺ If the sequence has paused at an editable record, choose **Edit** to edit the record.

Choose **Help** to get information on the current error message.
- ❻ This area shows the current error message, if any. Clicking the Current Error panel displays the error database help page for the current error, if any.
- ❼ These buttons have the same options for controlling sequence execution as they do for debugging a sequence. See "Debugging Sequences" on page 66.
- ❽ This area shows the current state of the runtime task indicated by ❸.

4.10 Editing Database Records During Execution

The debug options can be activated or deactivated at any time prior to or during execution of a sequence. To activate the editing function for location fields, select ☒ **Walk Thru** and ☒ **Edit All**.

The sequence will stop at the first editable record. A message similar to:

Proceed or edit: loc1 locations: loc.rec.name

is displayed. To edit the record, choose **Edit**; MotionWare will enter record editing mode with the proper record displayed. (Note: Not all fields in a record are editable during debug execution.) After you have completed editing of the record, **Go ➡ Back** from the record editing menu (or press SHIFT+F3), and you will be returned to the Task Control Panel.

To execute the statement containing the edited record, choose the **Proceed** push button to execute the statement. To re-execute the statement, choose the **Retry statement** push button. These operations can be repeated until the record is properly edited.

¹ See the description of the **mowini** initialization databases (Table A-3) for details on how to change the module name associated with the Master Control Panel.

When you have completed editing of a given record, choose the **Proceed** push button to move to the next record in the sequence. If the sequence stops at a record you do not want to edit, choose the **Proceed** push button to continue execution.

To return to normal sequence execution, deactivate the ☒ **Walk Thru** and ☒ **Edit All** options to continue normal execution.

An executing sequence can be viewed in the sequence editor by:

Edit ➡ Sequence

The sequence displayed will be in read-only mode (read only will be indicated on the status line at the bottom of the sequence editor). You will not be able to add, delete, or change statements or statement arguments.

Activating Single-Step Execution

It is often useful to execute an application one step at a time to discover where problems are occurring. To single-step through a MotionWare sequence, select ☒ **Statement** in the Pause After group to stop execution at the selected point. To resume or abort execution, choose one of the push buttons at the bottom of the Task Control Panel.

To resume normal execution, deactivate any options selected from the Pause-After group.

Location Databases

5

5.1 Editing Locations	74
Location Record Options	74
Example Strategy Routines	78
Optimized Moves	78
5.2 Using Pallets.....	80
Pallet Parameters.....	81
How Rows and Columns Are Determined	81

5.1 Editing Locations

Each robot location is associated with a record in a location database. Location values can be entered manually, recorded from the current robot location, or taught using the manual control pendant. They also can be created from CAD data by using the ADX utility.

To use a location database, the sequence and the location database must be in the same Resource Module. (All sequences in a resource module use the same location database.) Location databases are created from the resource module utilities (see Chapter 3). To edit records in a location database, the associated resource module must be loaded and selected.

Once a location database has been created, the database location records are created and edited using the database editor.

To access location records:

Edit ➔ Locations

The following screen is displayed:

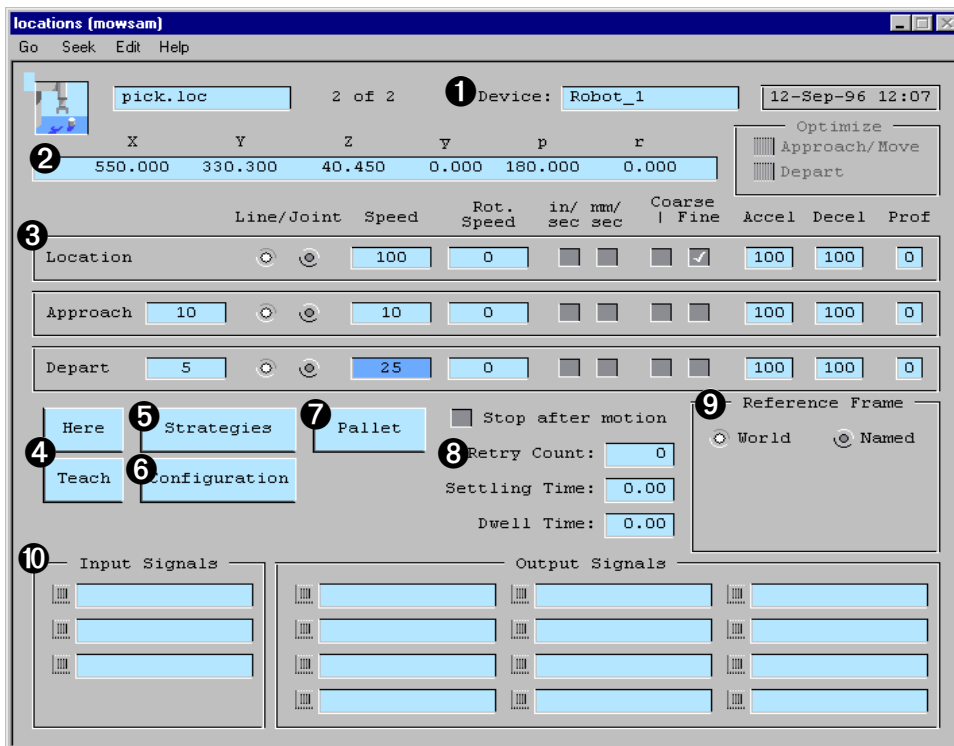


Figure 5-1
Location Database Menu Page

Location Record Options

To create a new location, press the NEW key (F2). To edit existing locations in this database, use the Seek pull-down options to display different location records. The location database you are editing and the associated resource module are shown in the title bar.

- ❶ A location record must be associated with a robot. Double click the Device data box to display a list of available robot types. Select the name of the device that corresponds

to the robot that you want to move to this location. If you attempt to move a different robot to this location, AIM will display an error message. In the unusual case in which you can move more than one robot to the same location, select “Any_Device” to disable this error checking.

- ② Displays the component values of this location. If you are very familiar with the way location components work, particularly with reference frames, you can edit the components individually. Normally, you would teach these values using either the **Here** or **Teach** buttons.

Optimized moves are available only if your system is equipped with the Adept HyperDrive option. See “Optimized Moves” on page 78 for details.

- ③ Set the motion parameters for the robot to move to this location. Motion parameters are described in section 2.2 on page 30.

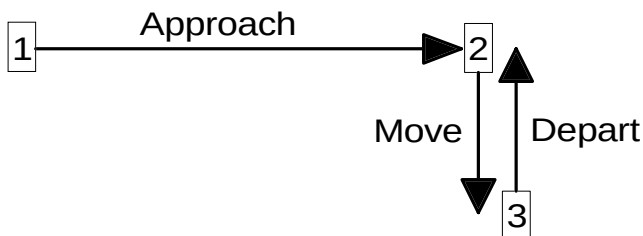
If you want the robot to approach this location from a point directly above the location,¹ enter the height value in the Approach data box. The motion parameters specified with the approach height will be in effect for the move from the previous robot location to the approach location. The motion parameters specified for the location will be in effect for the move from the approach location to the location. Be careful not to specify approach heights that are unreachable by the robot.

If you want the robot to move straight up¹ before moving to the next location, enter the depart height in the Depart data box. The motion parameters specified will be in effect for the move from the location to the depart height. Be careful not to specify depart heights that are unreachable by the robot.

- ④ Choose **Here** to record the present robot location in this record.

Choose **Teach** to use the manual control pendant to teach the location for this record.

- ⑤ Displays the Location Strategies menu page (see Figure 5-3). You can specify a V+ routine or an AIM sequence to run at the beginning of the approach move, at the beginning of the move to the actual location, and at the beginning of the depart move. You also can specify a force process at the location (when force sensing is enabled). See Chapter 11 for details on the Force Sensing Module.



1. Approach strategy sequence called here.
2. Move (location) strategy sequence called here.
3. Depart strategy sequence called here.

Figure 5-2
Using Strategy Sequences

¹ Approach and depart move along the tool Z-axis.

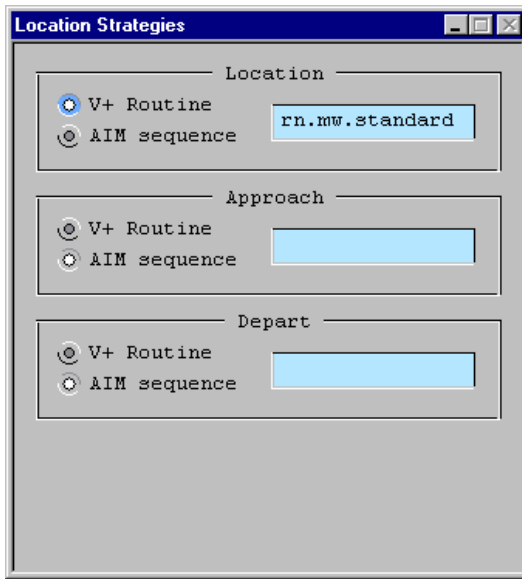


Figure 5-3
Location Strategies Menu Page

The standard location V^+ routine is **rn.mw.standard**. It performs the following operations:

- After reaching the location, wait for the time specified in settling time.
- Cycle the end effector. The first three output signals are set to the indicated states when the robot reaches the location (any output signals specified as 0 are ignored).
- Wait for the specified dwell time.
- Check the first input signal (if defined) and pause the sequence if it is not in the indicated state.

- ⑥ Choose **Configuration** to select the desired arm configuration options for this move.

The following screen is displayed:

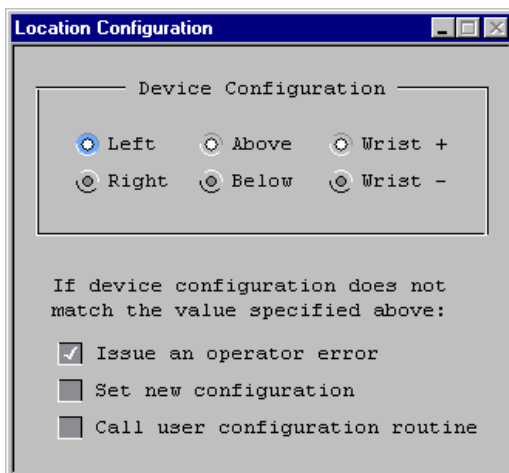


Figure 5-4
Location Configuration Menu Page

The configuration options available will depend on the type of robot you are using. The left/right option for Adept and other SCARA-type robots is described in Chapter 2. For details on the arm configuration options see the device module documents for other types of robots. See “Arm Configuration” on page 33.

- 7 Choose **Pallet** to define this location as a pallet. Pallets are described in section 5.2.
- 8 If a location strategy is being used that allows multiple attempts to place a part, the Retry Count data box specifies the maximum number of attempts that should be made.

If **rn.mw.standard** is used as a strategy routine, settling time specifies how long the robot should spend coming to a complete halt before activating the gripper.

If **rn.mw.standard** is used as a strategy routine, dwell time indicates how long a robot should wait after it has come to a stop and activated the gripper.

☒ **Stop robot after motion**

When selected, the robot will come to a complete stop before moving to the next location. If the robot was tracking a moving conveyor, it will stop at this point.

- 9 Select a reference frame for this location:

☐ **World** is the default reference frame and has its origin at the robot base.

☐ **Named** uses any reference frame from the Frame database. (A data box is displayed requesting the name of a reference frame.)

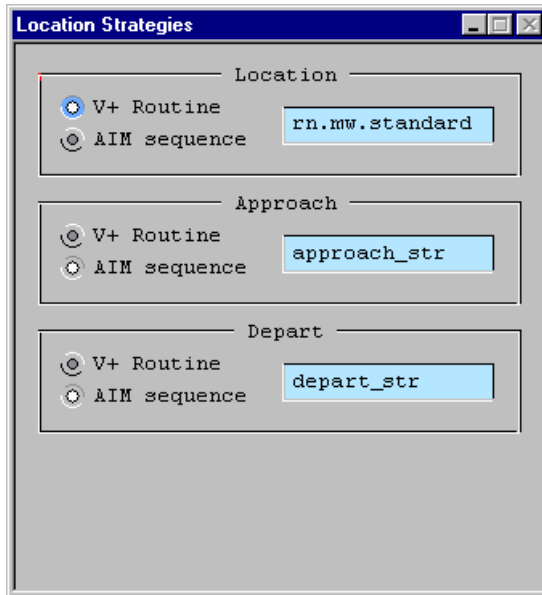
If a named reference frame is selected and the frame is based on conveyor data,

☒ **Keep if success** will cause the system to retain the frame values after a successful statement execution. ☒ **Keep if error** will cause the system to retain the frame values if an error occurs.

- 10 Specify the I/O signals that will be used by the strategy routines.

Example Strategy Routines

The sample Location Strategies menu page specifies the standard V⁺ routine **rn.mw.standard**, an Approach strategy sequence (**approach_str**), and a Depart strategy sequence (**depart_str**). These strategies are initiated at the points indicated in Figure 5-2.



The following example strategy routine uses the WAIT_UNTL_ROBOT statement to cycle the digital output when the robot is within 50mm of the pick location.

```

1.; APPROACH_STR
2.;
3.; This strategy sequence uses WAIT_UNTL_ROBOT. It
4.; forces the sequence to wait until the robot is within
5.; 50mm of the pick location before initiating the IO signals.
6.;
7.    WAIT_UNTL_ROBOT < 50 FROM pick
8.    IO OUTPUT 3001
9.    IO OUTPUT -3002
10.   WAIT 100
11.   IO OUTPUT -3001
12.   IO OUTPUT 3002
13.   WAIT 300
14.   IO OUTPUT 3001
15.   IO OUTPUT -3002

```

The following example strategy routine uses the MOVE_REL statement to rotate 90 degrees about the Z-axis during the depart move.

```

1.; DEPART_STR
2.;
3.; This strategy sequence uses MOVE_REL to rotate 90 degrees
4.; about the Z axis using a speed of 2 for the rotation.
5.;
6.    MOVE_REL WRT depart RZ 90 SPEED 2

```

Optimized Moves

To use the optimize option, the robot must be equipped with the Adept HyperDrive option. Also, the HyperDrive tuning utility must have been run to generate an optimal-

speed trajectory model for the robot. See the *Adept HyperDrive User's Guide* for details on the HyperDrive option. See the *Instructions for Adept Utility Programs* for details on running the tuning utility..

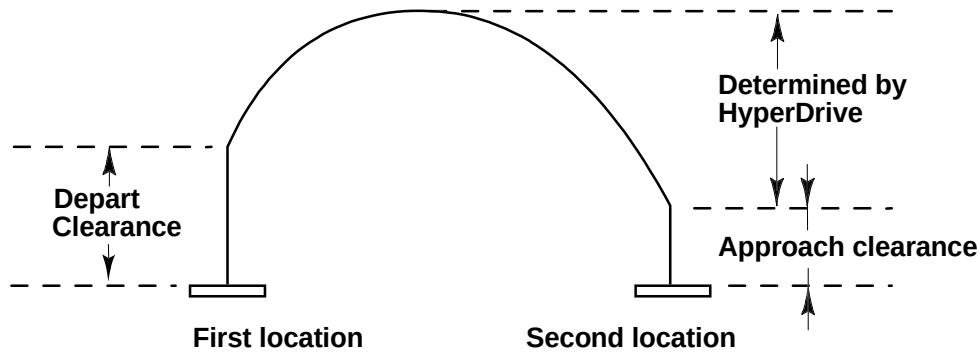


Figure 5-5
Elements of an Optimized Move

When the robot makes an optimized move (see Figure 5-5), the system determines the optimal trajectory for a depart-move-approach sequence. Therefore, a location with ☒ **Depart** selected must be followed by a move with ☒ **Approach/Move** selected.

When ☒ **Approach/Move** and ☒ **Depart** are selected, the top section of a location record appears as:

Figure 5-6
Optimized Approach/Move

In Approach Clearance, specify the minimum height the robot must reach when moving to the location. To optimize the trajectory, the robot may approach from a height higher than the recommended clearance. Optimized moves are not “square” moves, i.e., the optimum move height may exceed the approach and depart clearance heights (see Figure 5-5).

In the Model data box, specify the robot model that was created with the tuning utility. A different model must be created for each robot payload.

In the Accel/Decel data boxes (in Figure 5-1), specify the maximum torques the robot may apply during the acceleration and deceleration phases of the optimized move. The maximum value is 100%. A typical value is 80%. The lower the torque value, the closer the robot move will be to a “square” move. Torque values are scaled by the robot speed set at the control panels. If 80 is specified as the torque value for a move and 50 is specified as the control panel speed, the maximum torque applied will be 40%.

In the Depart Clearance data box, enter the minimum depart clearance and the maximum vertical acceleration torque. A move with this option selected must be followed by a move with ☒ **Approach/Move** selected. Otherwise, sequence execution will halt. (See Figure 5-5.)

In the Maximum vertical torque data box, enter the maximum torque allowed across the depart clearance portion of the move

5.2 Using Pallets

If parts are being acquired or placed at evenly spaced locations on a pallet, the pallet parameters allow you to tell the robot how many rows and columns are on a pallet and how far apart those rows are. You then have to create a location record only for the first pallet location.

To specify pallet parameters, choose the **Pallet** button (see Figure 5-1). The following screen is displayed:

The screenshot shows the 'Pallet Parameters' dialog box. It contains the following elements:

- 1** Use Pallet: ☐ No ☒ Yes
- 2** Frame:
- 3** Spacing
- 4** Count
- 5** Index
- 6** Done Signal
- Row:
- Column:
- Layer:
- 7** Index Update Ordering

	First	Second	Third
☒ Row	Column	Layer	
☐ Column	Row	Layer	
☐ Row	Layer	Column	
☐ Column	Layer	Row	
☐ Layer	Row	Column	
☐ Layer	Column	Row	
☐ Freeze all indices			
- 8**
 - ☐ Update if error
 - ☐ Reset if error
 - ☐ Freeze if error

Figure 5-7
Pallet Parameters

Pallet Parameters

- ❶ Indicate that the pallet parameters should be used at this location. Selecting ☒ **No** will cause MotionWare to treat this as a simple location regardless of how the pallet parameters are defined.
- ❷ If the location is relative to a named reference frame, that reference frame name will appear here and all locations on the pallet will be relative to that frame.
- ❸ Enter the space between individual part locations in the row, column, and layer direction. For a two-dimensional pallet, set the layer spacing equal to 0 and the count equal to 1.
- ❹ Enter the number of part locations in an individual row, individual column, and individual layer.
- ❺ Shows the column, row, and layer that the robot will access when the next part is picked up from this pallet. These fields are updated as the pallet is accessed. When all index values equal the count values, the last part will either be acquired or placed, and the index values will be reset to 1.

You can edit these values if you want to begin or continue filling/emptying a pallet at a different location.

- ❻ Specify an output signal that will be set to the indicated state when the row, column, or layer is complete.
- ❼ This section allows you to specify the order in which the robot will access the pallet. The last selection allows you to defeat update of the index counters for debug purposes.
- ❽ Normally, the pallet indices are reset to 1,1,1 when all rows, columns, and layers have been accessed.

If ☒ **Reset if error** is selected, the pallet indices will be set to 1,1,1 after an error occurs during the approach move.

If ☒ **Update if error** is selected, the pallet indices will be updated to the next location on the pallet even if an error occurs during the approach move.

If ☒ **Freeze if error** is selected, the indices are not changed if an error occurs.

If **Reset if error** and **Update if error** are both selected, **Reset if error** takes precedence.

How Rows and Columns Are Determined

Pallet rows are considered to be parallel to the reference-frame X-axis. Columns are parallel to the reference-frame Y-axis. Layers are parallel to the reference-frame Z-axis. This means that if the world coordinate system is used for the location, the pallet will have to be lined up exactly with the axes of the world coordinate system. Since this could be clumsy to set up, pallets normally use calculated reference frames. Reference frames are covered in Chapter 8.

The strategy for using a pallet feeder is:

- Create a reference frame based on three locations on the pallet

- Declare the location record to be relative to this reference frame
- Teach row 1, column 1, layer 1 of the pallet as the location

When the robot accesses the pallet, it will start at row 1, column 1, layer 1 and add the value specified in Spacing to each successive access until the number of accesses specified in Count has been made. Depending on which order is selected, the robot will then return to the original offset and add the value specified in Spacing for the next access.

Variable Database

6

6.1 Introduction	84
6.2 Accessing the Variable Database	84
Variable Database Record Options	84
A Note on Data Conversion	87
SELECT_VAR_REC Statement	87

6.1 Introduction

Many AIM statements can use values from the variable database to complete their arguments. For example, the number of cycles that a REPEAT statement executes can be determined by a variable. There is a global variable database, and there can be a variable database in the currently selected resource module.

6.2 Accessing the Variable Database

To access the variable database:

Edit ➡ Variables

The following screen is displayed:

Variables (mowsam)
Go Seek Edit Help

1 2 Group: Date:
1 of 1

Description

3 Variable to store to current state of the conveyor sensor. This record will be sorted with the "cnv" group.

4 Value:

5 Data Type
constant - Output signal
Define

6 Variable Scope
☒ Local to module
☐ Local to sequence
mowsam

7 DDE Access
☒ Read
☐ Write
Polling time
 sec

8 Definition
☒ Read/Write ☐ Read only
Output signal number:

Figure 6-1
Variable Database Menu Page

Variable Database Record Options

- 1 Enter a name string for the variable or symbolic constant to be defined. Enter a real number to define a real constant
- 2 Indicate whether this record should belong to a group. This option is for convenience in sorting variables in the various pick lists (variables are sorted first by Group and then by record name). It does not affect the behavior of the variable.

NOTE: The Variable Index pop-up shows all local-to-sequence variables. It also displays the sequence and group names with lines between each group for clarity.

- ③ Enter a description of the variable, what it does, when it is changed, etc.
- ④ Displays the value that corresponds to the information entered in item ③. For some selections in item ⑤, this field is used to enter a value for the variable. Question marks indicate that the variable has not been defined.
- ⑤ Define the Data Type for this record. The default data type is **numeric - AIM variable**. Choose **Define** to define a different data type.

Select **constant - Integer** to define a symbolic constant value that can have only an integer value, e.g., 112.

Select **constant - Input signal** to define an input signal. This value must represent a valid, installed digital input signal, e.g., 1003 (or a soft signal).

Select **constant - Output signal** to define an output signal. This value must represent a valid, installed digital output signal, e.g., 7 (or a soft signal).

Select **constant - Real value** to define a symbolic constant value that can have a real value, e.g., 12.635.

Select **numeric - AIM variable** to define a real variable that can be changed by an executing AIM sequence.

Select **numeric - ai.ctl[] value** to define a real variable that accesses the special AIM control values. See the *AIM Customizer's Reference Guide*.

Select **numeric - Database field** to define a real variable that refers to a database field that is a numeric type—byte, integer, or real.

The database field feature allows an AIM variable to be associated with a field in another database. When the variable is accessed, the specified database field is actually read from or written to. You can use the fields in item ③ to specify the database record. If these fields are left blank, the SELECT_VAR_REC statement must be used to specify the record.

Select **numeric - V+ variable** to define a real variable that can be changed by an executing V+ program. The value is stored in a V+ global variable.

Select **numeric - Vision result** to define a real variable that is associated with a vision inspection record result. It is a read-only type. The results of any vision inspection may be specified and then processed in a sequence program.

NOTE: You must supply the Vision record name in the Record field or a linking error will be reported.

Select **string - AIM variable** to define a variable that contains a string value. The value is stored in the AIM variable database. The string may be up to 128 bytes long. String variables may be used in the SETS statement or the I/O server statements.

Select **string - Database field** to define a string variable that refers to a database field that is a string type—string or name. See the paragraph under **numeric - Database field** for more details.

Select **string - V+ variable** to define a variable that contains a string value. The value is stored in a V+global string variable. The string may be up to 128 bytes long. String variables may be used in the SETS statement or the I/O server statements.

- 6 This area defines the sequences that will have access to the variable (known as variable scope). If the record being accessed is in a variable database which is part of a resource module, the variable can be:

☒ **Local to module** Variable accessible by all sequences in the module.

☐ **Local to sequence** Variable accessible only by a specific sequence.

- 7 Select ☒ **Read** to allow DDE read access to the variable information by an external Windows™ application such as Microsoft Excel.™

Select ☒ **Write** to allow DDE write access to the variable information by an external Windows™ application such as Microsoft Excel.™

Use the Polling time field to specify the polling time (in seconds) for the DDE operation.

- 8 The following Definition items are displayed based on the Data Type selected in item 5:

☒ **Read/write** Allows read/write access to the variable record.

☐ **Read only** Allows read-only access to the variable record.

Output signal number The output signal number (used when **constant - Output signal** is selected).

☒ **Global only** Refers to the Database field.

Database: The type of the database accessed by a numeric or string valued Database field variable.

If ☒ **Global only** is selected, the global database of this type is always accessed.

If ☐ **Global only** is not selected, the database of this type in the resource module is accessed. If there is a database of this type in the resource module, the global database is accessed.

Record: The record number to access in the specified database. If left blank, it accesses the last record selected by the SELECT_VAR_REC statement. See the "SELECT_VAR_REC Statement" on page 87 for details.

Field: The field name to access in the specified database.

Index: The index number that corresponds to the specified field name in the database.

Result:	For numeric - Vision result only. Displays the result of the specified vision record. You must supply the vision record name (in the Record field) or a linking error will be reported.
Value:	For constants and input/output signals, enter the value for this variable. This value cannot be changed except by editing it from the variable record. For variables and ai.ctll[] values, displays the current value of this variable. Question marks indicate that the variable has not been defined.

A Note on Data Conversion

In most places, AIM allows you to specify string or numeric values interchangeably. If you use a string-type variable where AIM expects a numeric value, AIM uses the V+ VAL() function to extract the leading number from the string and uses that value in the computation. If there is no number at the start of the string, AIM uses zero. If you use a numeric value where AIM expects a string, AIM uses the V+default numeric encoding (/D format) to create a string value (except that there is no leading blank in the string). The computation continues with that string. This feature is illustrated in the following example:

Assume we have the numeric variables **n1**, **n2**, and **n3**, and the string variable **s1**. You can write a sequence to set **n3** equal to the numeric value 12:

```
SET n1 = 1
SET n2 = 2
SETS s1 = n1+n2 ;Concatenate the string
SET n3 = s1      ;Convert the string value to a numeric value
```

SELECT_VAR_REC Statement

This statement dynamically selects a database record for use by the variable database for the current runtime task.

The syntax is:

```
SELECT_VAR_REC --database-- {GLOBAL --yes/no--} RECORD
--string_var-- {STATUS --o_variable--}
```

Double click --database-- to enter a standard database name for a runtime database. If GLOBAL is set to "yes" the global database of the specified type is used, even if there is a database of this type in the current module.

RECORD --string_var-- is the name of a string variable that contains the name of the record to be found. If the record is found, STATUS --o_variable-- is set to TRUE; otherwise, it is FALSE.

Note that a different record may be selected for each database type. The selection of each runtime task is independent of all others.

Paths 7

7.1 What Are Paths?	90
7.2 Creating Paths	91
Path Menu Page Options	92
7.3 Editing Path Segments	94
Path Segment Menu Page Options	94

7.1 What Are Paths?

Paths are predefined “safe corridors” used by the robot when moving from one location to another. Predicting the actual path of the robot becomes difficult if you are picking up parts from multiple locations, placing parts on assemblies in different locations, building multiple assemblies in the same workcell, or using pallet feeders.

In a crowded workcell where there are many obstructions that the robot must avoid, many opportunities for unpredictable robot motion exist. This is neither desirable nor acceptable. To overcome the problem of unpredictable robot motions, you create paths, which are collections of locations that you know the robot can safely move through.

In addition to specifying the location points on a path and the motion parameters associated with each location point, you specify whether each location point is an exit and/or entry point on the path. The robot can enter a path only at an entry point and leave it only at an exit point.

When a clause in a statement instructs the robot to move to a location along a path, MotionWare will calculate the nearest entry point on the path and move the robot to that location (using the motion parameters specified for the entry point). MotionWare will then calculate the exit path nearest the location being moved to, move the robot through all points on the path between the entry and the exit point, and move from the exit point to the location. All motion parameters can be set for each point in a path (see “Motion Parameters” on page 30).

Figure 7-1 shows an example of an “along” path used to avoid an obstruction in the middle of the workcell. The robot can enter the path from either feeder and must pass through both transit locations. It can exit at any one of the three exit locations.

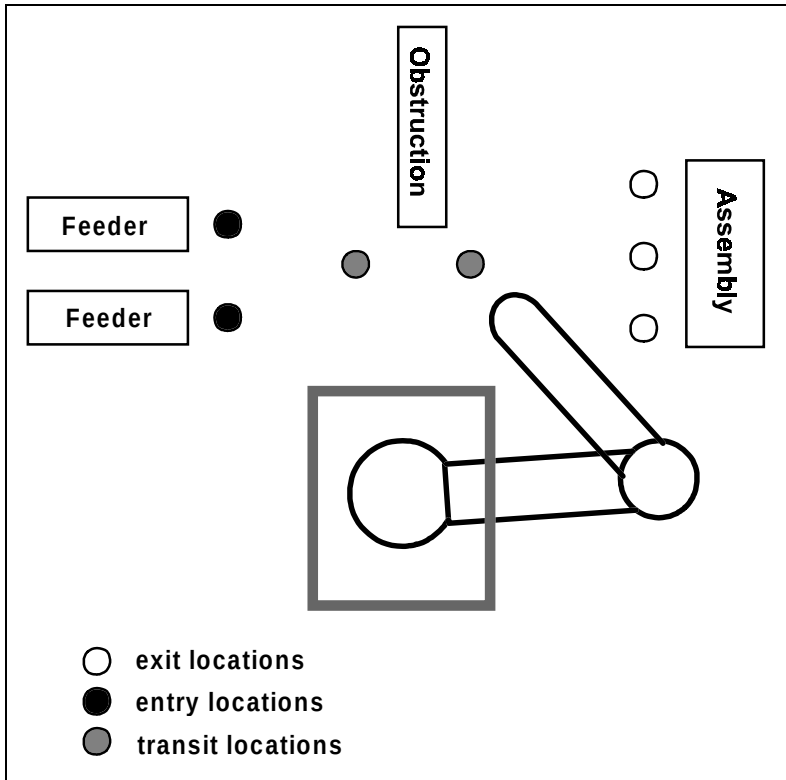


Figure 7-1
Example of an “Along” Path

7.2 Creating Paths

To create a new path:

Edit ➔ **Path** ➔ **New Path** ➔ *enter path name* ➔ **Create**

To edit an existing path:

Edit ➔ **Path** ➔ **Seek** ➔ **Index** ➔ *double-click path*

The following screen is displayed:

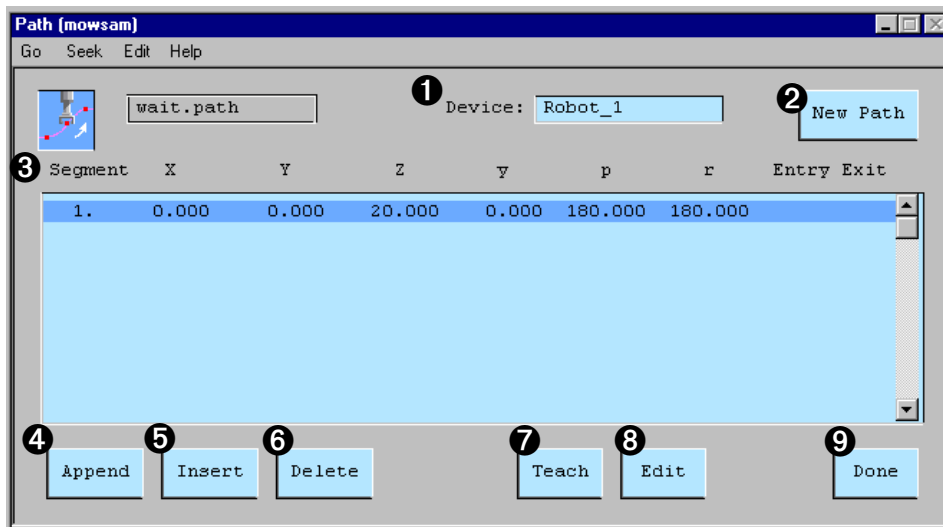


Figure 7-2
Path Menu Page

Path Menu Page Options

- ❶ Select the device that you want associated with this path. Double click the Device data box to display a list of devices available to your system. The actual device names will depend on the particular motion devices installed on your system.

To choose a different path to edit:

Seek ➡ Index ➡ *double-click path*

- ❷ Choose this button to create a new path. A window is displayed prompting a name for the new path. Enter a new name and device number.

Choose **Create** to create a new path.

- ❸ Shows all locations (segments) on the path. A check mark in the Entry column indicates that this is a point at which the robot will be allowed to enter the path. A check mark in the Exit column indicates that this is an exit point from which the robot will be allowed to leave the path. If neither column is checked, the point is a transition point. This means that the robot must move through the point, but it can neither enter nor leave the path from a transition point.

The selected path segment (dark blue bar) is the current location (or path segment). Choosing any of the buttons on the bottom of the menu page will affect the selected path segment.

- ❹ Choose this button to add a new segment to the end of the path.
- ❺ Choose this button to add a new segment above the selected segment.
- ❻ Choose this button to delete the selected segment. If there is only one segment in the path, choosing this button also will delete the path.
- ❼ Choose this button to enter the manual control pendant teaching mode to teach or update the selected segment.

- ⑧ Choose this button to display the editing screen (see Figure 5-5) for the selected segment. (This button is equivalent to double-clicking a segment.)
- ⑨ Choose this button to exit path editing.

7.3 Editing Path Segments

To create a new path segment, choose either the **Append** or **Insert** button from the Path menu page (see Figure 7-2).

To edit an existing segment, click the segment (in the pick list shown in Figure 6-2) to select it and either choose **Edit** or double click the segment. The following screen is displayed:

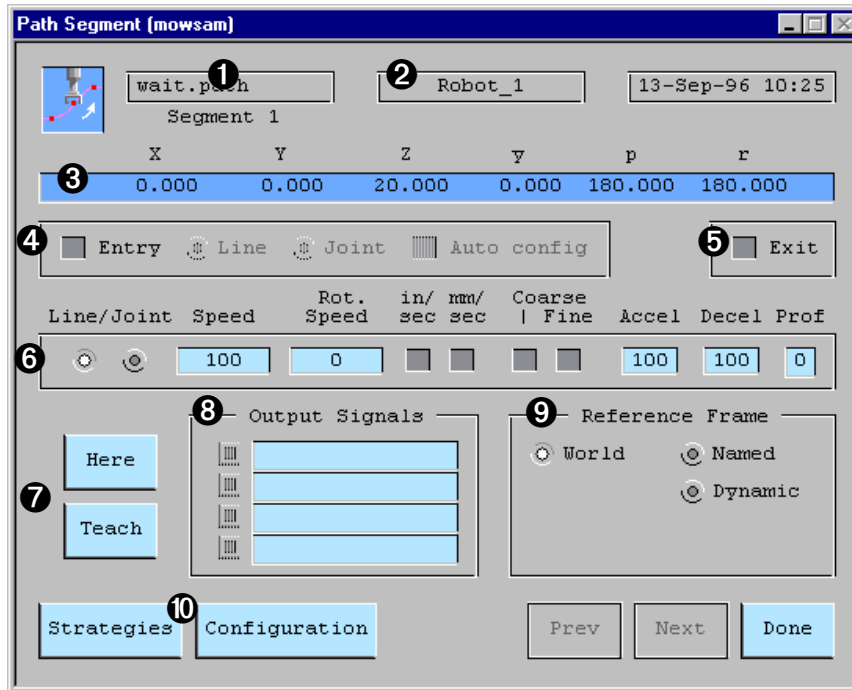


Figure 7-3
Path Segment Menu Page

Path Segment Menu Page Options

- 1** Shows the associated path name (class) for this segment and the number of this segment (1).
- 2** Shows the motion device specified in the path record (item **1** on page 92). The date and time indicate when this segment was last modified.
- 3** Shows location values for this segment.
- 4** Select this option if this segment is to be an entry point.

Select the appropriate radio button to indicate whether the motion from the previous robot location to the entry point should be a straight-line motion or joint-interpolated motion. This setting will be in effect for the move from the robot's previous location to this entry point on the path. The settings in item **6** will be used when this point is a transition point.

- 5** Select this option if this point is to be an exit point from the path.

- ⑥ Select the motion parameters for the robot motion to this point on the path. If the robot is entering the path at this point, the parameters selected in item ④ will override these parameters. Path segments do not use approach or depart heights.

- ⑦ Choose **Here** to make the current robot location the location for this path segment.

Choose **Teach** to use the manual control pendant to teach this location. See the *Manual Control Pendant User's Guide*.

- ⑧ If a strategy routine is specified in item ⑩ and the strategy routine uses any of the output signals, specify the output signal number here. These data boxes can also reference a digital I/O record from the variable database. Double click the individual data boxes to display a list of records in the variable database.

If no V+ strategy routine or AIM strategy sequence is specified for a path segment, the output signals defined for that segment are asserted as soon as the robot begins moving toward the indicated path point.

The signals are asserted even if the robot is moving backward along the path.

The location parameters specified in items ④, ⑤, ⑥, and ⑦ are described in "Motion Parameters" on page 30.

- ⑨ Indicate the reference frame this location should be relative to (see section 8.1 and section 8.2 for a discussion of reference frames):

- **World** The default reference frame (origin at the robot base).
- **Named** This option is used only for certain vision options and is described in Part Two of this manual.
- **Dynamic** Not used in standard MotionWare. Used for some PCB statements.

- ⑩ Choose **Strategies** to specify a V+ program or AIM sequence to execute when the path segment is reached.

Choose **Configuration** to specify the arm configuration options for this segment of the path.

Choose either **Prev** or **Next** to edit the previous or next segment in this path.

Choose **Done** to return to the path menu page (Figure 7-2).

Reference Frames 8

8.1 What Is a Reference Frame?	98
8.2 Creating Reference Frames	100
Frame Menu Options	100
8.3 Tool Transformations	102
Creating a Tool Transformation	103
Current Tool	104
Tools and Statements	104

8.1 What Is a Reference Frame?

A MotionWare location is specified as an offset from a given point. For a default (world) location, the offsets are from the origin of the world coordinate system (base of the robot).

Figure 8-1 shows the world coordinate system for an Adept SCARA robot.

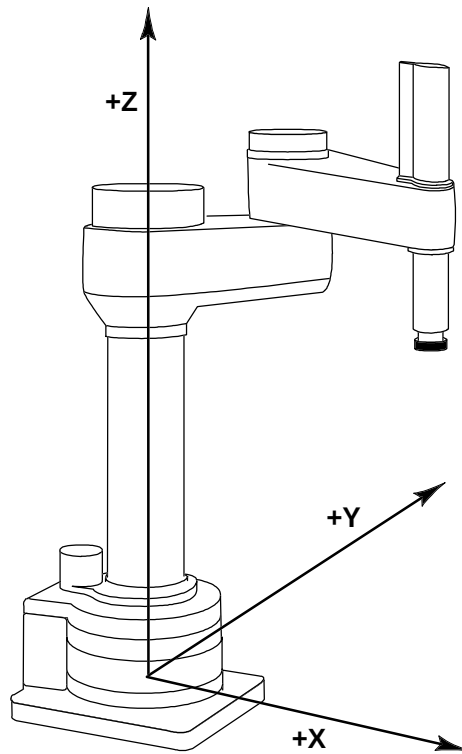


Figure 8-1
World Coordinate System

If, after you set up your workcell, nothing ever moved, this coordinate system would be sufficient. In fact, you may move part feeders, the location of assemblies you are building may change, or the robot itself may need to be moved or recalibrated. In any of these cases, retraining the locations in your workcell will be easier if the majority of location offsets are relative to a coordinate system other than the world system. Figure 8-2 shows such a situation.

The coordinate system in Figure 8-2 has its origin at the starting location of a pallet. The distance between pallet locations is 40 millimeters. Notice that the first location is offset from the reference frame by 40 millimeters in the Y-direction. Therefore, only the Y-component has a nonzero value.

The second location is offset from the reference frame in both the X- and Y-directions. Consequently, these two components have nonzero values. If this pallet were brought into the workcell at a different location, merely updating the reference frame to reflect its new location would allow you to use any locations created relative to that frame without further modifications.

Reference frames can be moved and rotated in three dimensions. In this example the reference frame is rotated exactly 90° from the world coordinate system. Note, however, that you could move a reference frame to a point where the locations relative to that frame would be out of range for the robot.

This example shows no rotation of the end effector. In fact, if this were an assembly with several different parts being placed with different orientations of the end effector, the orientation components of the location also would have nonzero values.

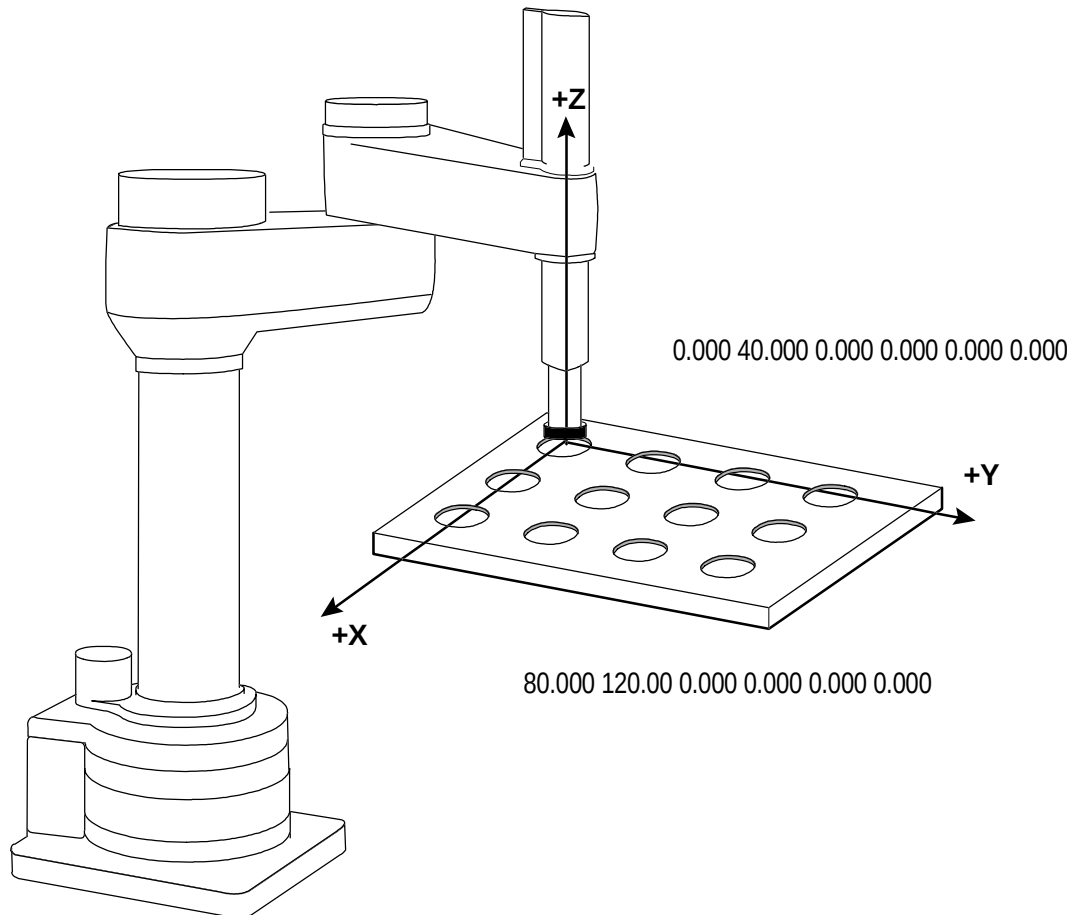


Figure 8-2
Locations Relative to a Reference Frame

8.2 Creating Reference Frames

Reference frames can be calculated from taught locations; in the case of moving (conveyor) frames, they are updated based on current conveyor status.

This section describes creating reference frames from taught locations. Chapter 12 describes using moving reference frames.

To create a new reference frame:

Edit ➔ Frame ➔ press the NEW key (F2)

To edit an existing frame:

Edit ➔ Frame ➔ Seek ➔ Index ➔ double click frame

The following screen is displayed:

Frame [mowsam]
Go Seek Edit Help

1 **new_frame** 2 of 2 2 ☒ Fixed ☐ Variable 3 13-Sep-96 10:43
☒ Object Definition Device: Robot_1

4 Description
Sample reference frame

5 Frame Value
Compute

	X	Y	Z	Y	p	r
	2.012	30.600	54.500	0.000	0.000	0.000

6 Frame Origin
Here Teach

	X	Y	Z	Y	p	r
	2.012	30.600	54.500	0.000	180.000	0.000

7 Point on +X axis
Here Teach

	X	Y	Z	Y	p	r
	53.100	30.600	54.500	0.000	180.000	0.000

8 Point near +Y axis
Here Teach

	X	Y	Z	Y	p	r
	10.123	76.554	54.500	0.000	180.000	0.000

Figure 8-3
Frame Menu Page

Frame Menu Options

- 1 Shows the name of this frame, the record number of this frame, and the total number of frames in the database.
- 2 Select ☒ **Fixed** to define a simple reference frame. The other options make this frame a moving frame for a conveyor belt. Chapter 12 describes nonvision moving frames. Chapter 15 details conveyor systems used with the vision option.
- 3 Select the device to be associated with this reference frame. The available options will depend on the motion devices installed on your system.
- 4 Enter a description of this frame.
- 5 Shows the value of the computed frame (once all the points have been taught and the **Compute** button has been chosen). The reference frame will be calculated based on

three points that you teach in the workcell. The origin of this frame will be the location in item **6**.

- The direction of the positive X-axis is determined by item **7**.
- The direction of the positive Y-axis is determined by item **8**.
- The direction of the Z-axis is determined by the relationship between items **7** and **8**.

If any of the locations change, choosing **Compute** will recalculate the reference frame. A yellow warning sign is displayed whenever the frame value requires updating (any time item **6**, **7**, or **8** has been changed).

- 6** Choose **Here** to use the present location of the robot as the reference frame origin (X, Y, and Z = 0).

Choose **Teach** to use the manual control pendant to teach a location to be used as the origin of the reference frame. Any locations that are made relative to this reference frame will use the X, Y, and Z values of this location as the starting point for their X-, Y-, and Z-offsets.

- 7** Choose **Here** to use the present location of the robot as a point on the positive X-axis of the frame. The reference frame X-axis will pass through the origin point specified in item **6** and this point.

Choose **Teach** to use the manual control pendant to teach a location on the positive X-axis.

- 8** Choose **Here** to use the present location of the robot as a point near the positive Y-axis of the frame. The reference frame Y-axis will pass through the origin point specified in item **6**, perpendicular to the X-axis line created from items **6** and **7**, and have its positive Y-direction pointing away from the X-axis line in the direction of this point.

Choose **Teach** to use the manual control pendant to teach a location in the positive Y-direction.

8.3 Tool Transformations

A tool transformation is a special transformation that accounts for robot grippers (or parts held in grippers) that are offset from the center of the robot tool flange. If, without a tool transformation, a location is taught using a part secured by an offset gripper, the actual location recorded will be the center of the tool flange that the offset gripper is attached to (see Figure 8-4).

If the same location is taught with a tool transformation in place, the location recorded will be the center of the gripper, not the center of the tool flange. This allows you to change grippers and still have the robot reach the correct location. Figure 8-4 shows the location of the robot when a location is taught and the actual location that is recorded. If the proper tool transformation is in effect when the location is taught, the location recorded will be the assembly location and not the center of the tool flange.

Tool transformations are most common when using CAD data to create assembly locations. If a tool transformation is not in place and the robot moves to a location generated by CAD data, the robot will move the center of the tool flange to the location instead of the center of the gripper. To correct this situation, the offset from the tool flange to the center of the gripper must be taken into account before a move to the CAD data-generated location is made. If a TOOL is defined for the gripper, MotionWare will automatically include the correct offset.

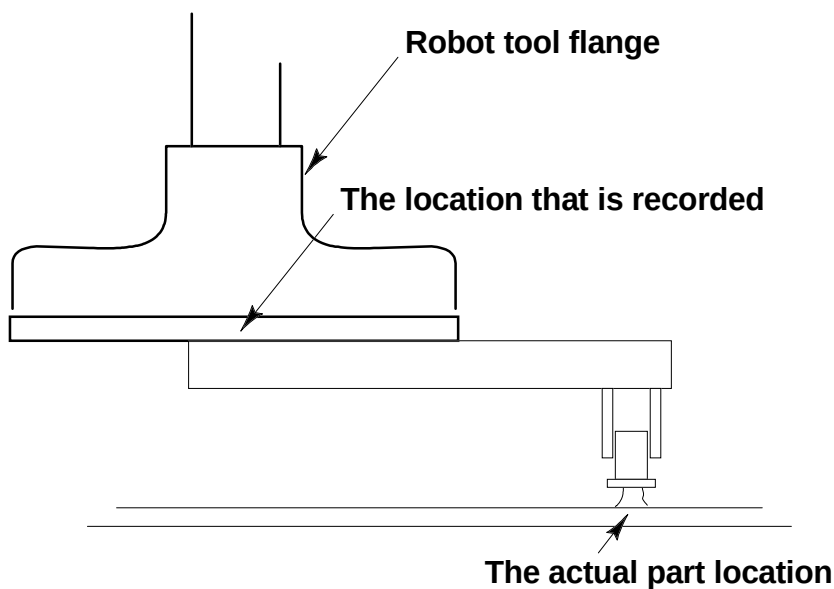


Figure 8-4
Tool Transformation Example

Creating a Tool Transformation

A tool transformation is the aggregate of the location components required to get from the center of the tool flange to the actual center of the gripper. The engineering drawings for your gripper should provide you with this information. (The *V⁺ Operating System User's Guide* contains instructions for teaching a tool transformation.)

To create a tool transformation:

Edit ➔ Tool ➔ press the NEW key (F2)

The following screen is displayed:

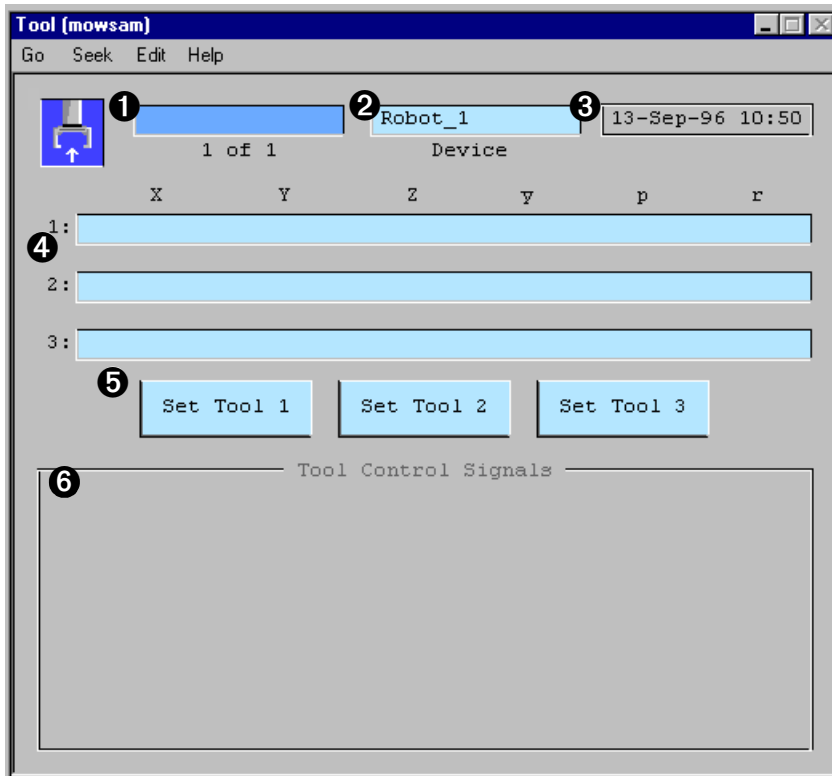


Figure 8-5
Tool Record

- ❶ Enter a name for the tool record. The total number of records in the Tool database (1 of 1, in the above figure) indicates the number of this record in the database and the total number of records for this tool.
- ❷ Double-click this data box to display a list of motion devices and select the one you want associated with this Tool record.
- ❸ Shows the date and time that this record was last modified.
- ❹ Enter the values for the tool offset. (When a sequence statement that has a tool argument is executed, the value of tool #1 is used.)
- ❺ Choose a **Set** button to use the values in item ❹ as the current tool.
- ❻ Tool Control Signals are not used in MotionWare.

Current Tool

A current tool can be set for all robot operations by choosing a **Set** button from the tool record (Figure 8-6).

The current tool also can be displayed or set by:

Setup ➔ Display/Change Tool

The following screen is displayed:

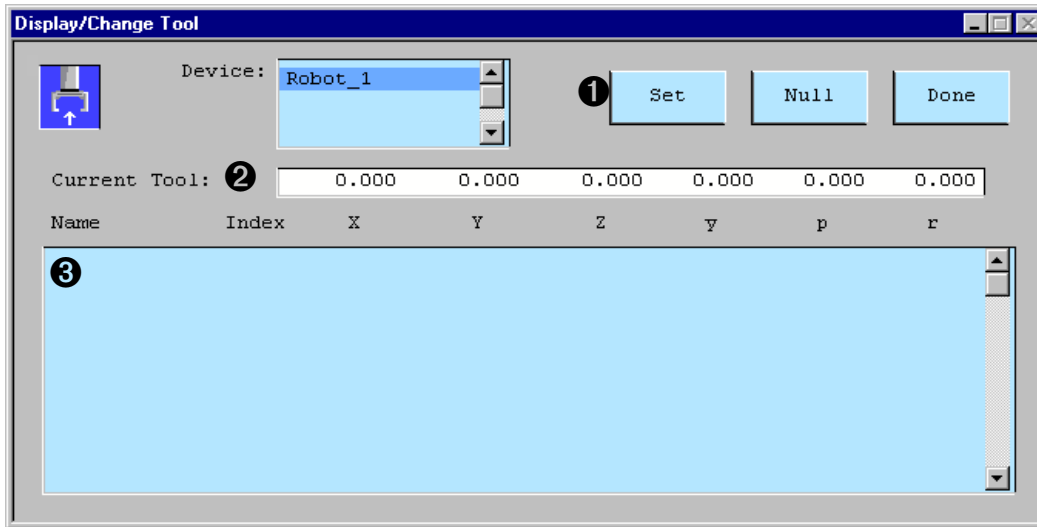


Figure 8-6
Display/Change Tool Menu Page

- ① Choose **Set** to make the selected tool values (item ③) the current tool.
Choose **Null** to reset the tool values to 0.
Choose **Done** to exit the tool display.
- ② Shows the values of the current tool. These values are not dynamically updated.
Choosing the REDRAW key (SHIFT+F6) will force an update of the current tool values.
- ③ Displays the values of all defined tools.



WARNING: Changing a tool to an unexpected value will cause the robot to move to incorrect locations.

Tools and Statements

If the optional {USING tool} clause of a statement (MOVE, for example) is specified, that tool is made the current tool. A tool transformation also can be set using the SET_TOOL statement.

Whenever a tool is changed by an executing sequence, that tool is in effect until changed by another statement or by selecting a new tool from the Display/Change Tool menu page (Figure 8-6).

Using Digital I/O 9

9.1 Introduction	106
9.2 Installing Digital I/O	106
Displaying the State of Digital I/O Signals	106
9.3 Digital I/O Signal Numbering	107
Soft Signals.	107
Special Signals	108
Negative Signal Numbers.	108
9.4 Using Digital I/O With MotionWare	108
Setting the State of Digital Signals	108
Waiting for a Digital Signal	108
Digital I/O and Conveyor Belts	109
Digital I/O and Locations.	109
OK_SIGNAL	109
Control Statements.	109
9.5 Using Digital I/O Statements.	109
WAIT_FOR Statement	109
WAIT_UNTIL Statement.	110
IO Statement.	110
IO_LIST Statement.	111
IO_LIST_VAR Statement	111

9.1 Introduction

Digital I/O provides you with a way of communicating with other devices in your workcell. Digital input allows your sequence to suspend or resume execution based on feedback from a switching device in the workcell. The state of digital input signals can be controlled by almost any type of switching device. Closing a microswitch when an object is in the correct location is a common use of digital input.

Digital output allows a sequence to signal devices in the workcell to begin or suspend operation. Digital output acts as a switch to open or close the current being supplied to the device. (Digital output switches the current: It does not provide it.) Common uses of digital output are controlling conveyor motors or part feeders and activating warning lights or sound devices.

9.2 Installing Digital I/O

Before a sequence can use the digital I/O capabilities, the following steps must be taken:

- Install the devices that generate or receive digital signals in the workcell.
- Install the physical digital I/O equipment in your controller. This installation is covered in the controller user's guide that came with your controller.

Your controller may not be equipped to use all available digital signals. To see which digital I/O signals have been installed in your system:

adept ➡ Monitor ➡ enter IO

The status of all digital I/O signals is displayed. The digit "1" indicates that the signal has been installed and is on. The digit "0" indicates that the signal is off. The character "-" indicates that the signal is not installed.

To terminate the digital I/O display, hold down the CONTROL key and type: c.

Displaying the State of Digital I/O Signals

The states of digital I/O signals can be set and/or viewed by:

I/O ➡ Digital Signals

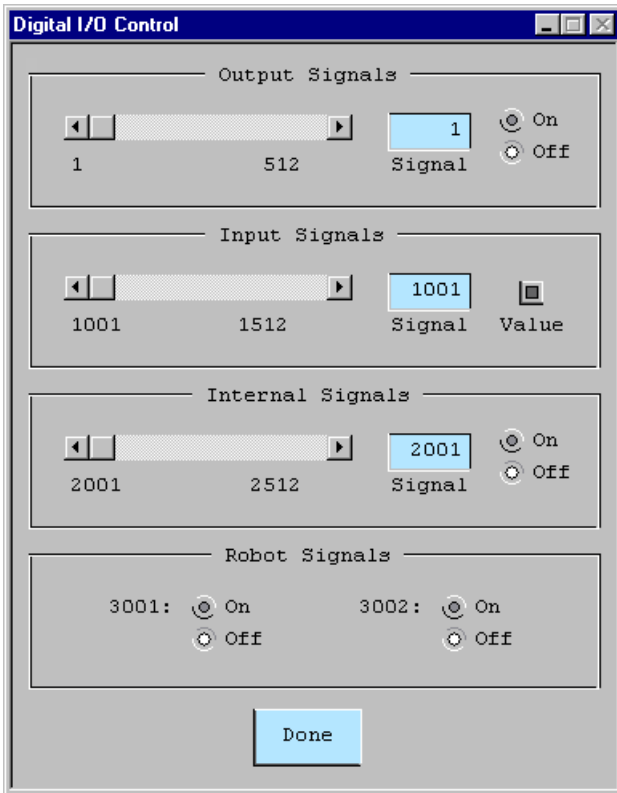


Figure 9-1
Digital I/O Control

This display will show only the signals that are installed in your controller. The input, output, and soft digital signal ranges are displayed. Either use the slide bar to select the desired digital signal or enter the desired signal value into the data box next to the slide bar. Click the on/off button to set the desired state of output or soft signals.

The digital signal will remain in this state until a sequence or program instruction changes the signal. If a signal has not been installed, you will not be able to select the signal number.

9.3 Digital I/O Signal Numbering

Digital input signals are differentiated from digital output signals by the range of numbers that each falls within; each range is limited to a maximum of 512 signals. Digital input signals must be in the range from 1001 to 1512, and digital output signals must be in the range from 1 to 512.¹

Soft Signals

There is an additional range of I/O signals from 2001 to 2512 that can be used with MotionWare. This range of signals, referred to as soft signals, is set by the system software and does not require hardware support. Soft signals are always available for use (although the range from 2005 to 2032 is reserved for use by AIM software). Soft signals can be used anywhere input or output signals are specified.

¹ Signals 9-31 and 1013-1031 are not available.

Special Signals

The range (3000-3008) performs special functions for Adept robots. See your robot user's guide for details on use of any of these special signals.

Negative Signal Numbers

When digital signals are set, they are set to a given state (not changed from the existing state). When MotionWare is instructed to set a positive output signal number, that signal is turned on. If the signal is already on, its state is not changed.

When MotionWare is instructed to set a negative output signal number, that signal is turned off. If the signal is already off, its state is not changed. Similarly, if MotionWare is instructed to wait for a positive input signal, it will suspend execution until the signal is on. If the signal is already on, execution will not be affected.

If MotionWare is instructed to wait for a negative input signal, it will wait until the signal is turned off. If the signal is already off, execution is not affected.

9.4 Using Digital I/O With MotionWare

MotionWare uses digital I/O to perform three classes of tasks:

- Monitor an input, output, or soft signal to see if it is in a given state (on/off, open/close, true/false).
- Set the state of an output or soft signal.
- Control the execution of statement blocks enclosed in conditional statements. See "General AIM Statements" on page 177.

Setting the State of Digital Signals

The state of digital output and soft signals can be set by:

IO ➡ Digital Signals

During sequence execution, digital output and soft signals can be set by the MotionWare statements. The MOVE statement allows you to set a digital signal when the statement has completed. A common use of this option in the MOVE statement would be to signal a conveyor belt to move a completed assembly out of the workcell.

The statements IO and WAIT_FOR can set a signal whenever they are encountered in a sequence.

Waiting for a Digital Signal

During sequence execution, a sequence can be paused until a digital signal reaches the desired state. A common use of this option would be to wait for a part-ready sensor to indicate that a part is ready before executing a statement.

The WAIT_FOR and IO statements allow you to either pause a sequence until a digital signal reaches the desired state or set a digital signal once a statement has executed.

Digital I/O and Conveyor Belts

Conveyor belts use digital I/O to cycle the belt, indicate that a belt is ready to be accessed, and determine that a part has passed a given point on the conveyor belt. See Chapter 12 and Chapter 15 for details on digital I/O with conveyor belts.

Digital I/O and Locations

Each location record allows you to specify digital output signals to set when the robot reaches the location. These signals are intended to activate the robot end effector. Digital input signals can also be monitored to indicate the success of the part acquisition or placement. These signals are controlled by the acquire/release strategy routine.

OK_SIGNAL

Many statements, including INSPECT and INSPECT_LIST have an OK_SIGNAL clause that allows you to set a signal based on the success or failure of the statement.

Control Statements

The control statements described in “General AIM Statements” on page 177 can be evaluated based on the state of digital I/O signals.

9.5 Using Digital I/O Statements

There are four statements that control digital I/O signals: WAIT_FOR, WAIT_UNTIL, IO_LIST, and IO.

WAIT_FOR Statement

The WAIT_FOR statement has one required and two optional arguments. The required argument is a constant or variable database record specifying a digital signal. When a WAIT_FOR statement is encountered in a sequence, the sequence will suspend until the specified signal achieves the correct state.

The syntax for WAIT_FOR is:

```
WAIT_FOR --input-- {OUTPUT --output-- {PULSE --constant--}}
```

If the statement:

```
WAIT_FOR dev_ready
```

is encountered in a sequence, the sequence will suspend until the digital signal specified in the variable database record “dev_ready” is turned off (if the signal is already off, the sequence will not wait).¹

The optional arguments to the WAIT_FOR statement allow you to specify a digital output signal to set and duration for output signal pulse.

If the statement:

```
WAIT_FOR dev_ready {OUTPUT dev_run {PULSE 10}}
```

¹ In all cases where a digital signal is expected, a minus (–) can precede the signal number or variable name. This will invert the sense of the signal. Thus “WAIT_FOR –1003” will wait until signal 1003 is low.

is encountered in a sequence, the sequence will suspend until the digital signal specified in "dev_ready" is turned on. When this digital signal is on, execution will resume and the digital output signal specified in the variable database record "dev_run" will be turned on for 0.1 seconds. (The constant replacing the PULSE argument is multiplied by 0.01 seconds to indicate the total duration of the signal change.) If a pulse argument is not entered, the specified digital output signal will remain on.

If the digital output signal is already at the indicated state, it is not changed.

WAIT_UNTIL Statement

The syntax for WAIT_UNTIL is:

```
WAIT_UNTIL {--uopr--} --variable-- {--opr-- --variable--  
{--opr-- --variable-- {opr-- --variable-- {--opr-- --variable--  
{--opr-- --variable--}}}} {TIMEOUT --constant--}
```

WAIT_UNTIL is similar to WAIT_FOR. However, it allows more complicated conditions to be evaluated. The statement has one required argument, a digital signal number or name from the variable database that specifies a digital signal number (--variable--). If only this argument is specified, sequence execution will halt until the signal is correctly set. The required --variable-- can be preceded with a keyword from the list of unary operators (this operation is applied after all clauses in the statement have been evaluated). NOT is the most commonly used unary operator.

Each optional --variable-- argument also takes a digital signal number or record from the variable database that specifies a digital signal number. Each signal can be preceded with a keyword from the list of operators. AND and OR are the most commonly used operators.

The final option argument in this statement is a TIMEOUT value. This value sets the maximum time (in 0.01 seconds units) that the sequence will pause waiting for the correct states of the specified digital I/O signals.

IO Statement

The IO statement has one required and two optional arguments. The required argument is a constant that requires a digital output signal number. When an IO statement is encountered in a sequence, the specified digital output signal will be set to the specified state.

The syntax for IO is:

```
IO {WAIT_FOR --input--} OUTPUT --output--  
{PULSE --constant--}
```

If the statement:

```
IO OUTPUT test.sig
```

is encountered in a sequence, the digital output signal specified in the variable "test.sig" will be turned on.

The optional arguments to the IO statement allow you to wait for an input signal before signaling the output signal and specify a duration for the state change of the output signal.

If the statement:

```
IO {WAIT_FOR -1012} OUTPUT -20 {PULSE 5}
```

is encountered, execution will suspend until digital input signal 1012 is turned off. When signal 1012 is turned off, digital output signal 20 will be turned off for 0.05 seconds. Execution will resume at the next statement.

IO_LIST Statement

This statement allows you to set up to 12 output signals at the same time.

The syntax is:

```
IO_LIST --output-- {--output-- {...}}}}}}}}}}
```

IO_LIST_VAR Statement

This statement allows you to set up to 12 output signals at the same time.

The syntax is:

```
IO_LIST_VAR variable_1 {variable_2  
                        {variable_3 {... variable_12}}}}}}}}}}}}
```

The IO_LIST_VAR statement is similar to the IO_LIST statement. However, the signal numbers are stored as AIM variables instead of constants.

Using the I/O Server **10**

10.1 Introduction	114
10.2 Installing the I/O Server	114
10.3 I/O Protocol Routines	114
10.4 Using I/O Server Statements	114
IO_OPEN Statement	115
IO_CLOSE Statement	115
IO_READ Statement	115
IO_WRITE Statement	116
10.5 Example I/O Server Sequence	117
10.6 Example Serial Driver Sequence	117

10.1 Introduction

The AIM I/O Server is designed to support multiple simultaneous I/O streams without blocking. The server is designed for convenient access to data files and serial devices such as bar-code readers. It is not designed for intensive high-bandwidth I/O operations such as file or image transfers.

The AIM I/O Server provides the following functionality:

- Allows access to standard V⁺ file-oriented devices (disks) and character-oriented devices (serial lines) from AIM sequences.
- Allows multiple AIM tasks to share an I/O resource. For example, multiple AIM tasks may write log data to a single disk file.
- Allows user-defined I/O protocol routines. These routines are written in V⁺ and loaded at AIM startup by an initialization database. These I/O routines may implement an arbitrary protocol and parse input or output data as required.

Additionally, AIM now supports a string data type for the variable database as well as a number of string operators (see Chapter 13 for details) that can be used to do data parsing. However, custom protocol routines provide more efficient operation.

10.2 Installing the I/O Server

The I/O server is part of the standard AIM Baseline software. It is installed as part of the process of installing MotionWare. See Appendix A for details.

To enable the I/O Server, you need to make a one-time change to the Baseline initialization database. After starting AIM, choose **Setup ➡ Initialization Data ➡ baseini.db** and set the value of the “I/O server enable” option to on.

10.3 I/O Protocol Routines

Adept supplies standard I/O protocol routines for:

- Variable-length sequential file I/O.
- Simple serial line I/O. Input records have control characters removed and are terminated by a carriage return. Output records are transmitted without interpretation or an additional terminator.

Adept also provides template routines for writing custom protocol routines. See the *AIM 3.1 Release Notes* for information on these routines.

10.4 Using I/O Server Statements

AIM I/O operations center around a “stream” ID that identifies an open I/O device. The stream ID is returned when an I/O device is open. It must be passed to any subsequent I/O statements that refer to the open device.

A “protocol” also is associated with each I/O device. The protocol identifies the method used to open and close the I/O device. It also specifies the format for reading data from or writing data to the device. (The protocol determines how communication is performed and how data is formatted.)

IO_OPEN Statement

The IO_OPEN statement is used to open an I/O device for subsequent I/O operations. The statement waits until the open operation is complete before returning a completion status code.

Multiple AIM tasks may access the same device. (The device is opened when the first task accesses the device.) To do this, the MODE argument must be the same for each AIM task that is accessing the device.

The syntax for IO_OPEN is:

```
IO_OPEN PROTOCOL --protocol-- IO_DEVICE {--s_uopr--}  
  --string_var-- {--s_opr-- --string_var-- {--s_opr--  
  --string_var-- {--s_opr-- --string_var--}}} MODE  
  --I/O_mode STREAM --o_variable-- {STATUS --o_variable--}
```

PROTOCOL is a keyword that is selected from a list of protocols defined in the I/O initialization database.

IO_DEVICE is a string expression using up to four string variables that specify the device, path, and filename as required by the selected protocol.

MODE is a keyword that is used by the protocol to determine the I/O open mode. This keyword is selected from a list of modes (read, write, or read_write).

STREAM is an output variable that receives an ID code used for subsequent I/O operations.

STATUS is an output variable that receives the I/O completion status code for the operation.

IO_CLOSE Statement

The IO_CLOSE statement is used to close the I/O device or file associated with an I/O stream and to free any associated system resources.

If multiple AIM tasks have opened this stream, the stream is not closed until the last task closes the stream. The last task waits until the close operation is complete before returning a completion status code.

NOTE: After this statement is executed, the stream ID should not be used for I/O by the task.

The syntax for IO_CLOSE is:

```
IO_CLOSE STREAM --variable-- {STATUS --o_variable--}
```

STREAM is a variable that specifies the I/O stream to be closed.

STATUS is an output variable that receives the I/O completion status code for the operation.

IO_READ Statement

The IO_READ statement is used to read data from an I/O stream while waiting for the specified time-out.

The syntax for IO_READ is:

```
IO_READ STREAM --variable-- {RECORD_NUMBER --variable--} {TIMEOUT
--variable--} DATA --o_variable-- {, --o_variable-- {,
--o_variable-- {, --o_variable-- {, --o_variable-- {,
--o_variable-- {, --o_variable-- {, --o_variable-- {,
--o_variable-- {, --o_variable--}}}}}}}} {STATUS --o_variable--}
```

STREAM is a variable that specifies an open I/O stream. Normally, the stream must be open for read or for read-write.

RECORD_NUMBER is used to specify the number of the record to be read. It is used only if it is supported by the I/O protocol.

TIMEOUT is used to specify the maximum number of seconds to wait for the read operation to complete. If this value is not specified, the statement waits indefinitely.

DATA is from 1 to 10 output variables that receive the data from the I/O device. The data format is specified by the protocol. The data is returned as strings, but is automatically converted to numeric data if numeric variables are specified.

STATUS is an output variable that receives the I/O completion status code for the operation.

IO_WRITE Statement

The IO_WRITE statement is used to write data to an I/O stream.

The syntax for IO_WRITE is:

```
IO_WRITE STREAM --variable-- {RECORD_NUMBER --variable--} DATA
--variable-- {, --variable-- {, --variable-- {, --variable-- {,
--variable-- {, --variable-- {, --variable-- {, --variable-- {,
--variable-- {, --variable--}}}}}}}} {STATUS --o_variable--}
```

STREAM is a variable that specifies an open I/O stream. Normally, the stream must be open for write or for read-write.

RECORD_NUMBER is used to specify the number of the record to be written. It is used only if it is supported by the I/O protocol.

DATA is from 1 to 10 output variables that contain the data to be written to the I/O device. The data format is specified by the protocol. The data is sent to the driver task as ASCII strings. If numeric variables are specified, they are automatically converted to strings using the V⁺ default (/D) format.

STATUS is an output variable that receives the I/O completion status code for the operation.

10.5 Example I/O Server Sequence

The example below shows how I/O Server statements can be used in an AIM sequence. This example creates the file "logfile.txt" on your default AIM directory. It then uses the IO_WRITE statement to write the data in the DATA parameter to the disk file.

```

I/O_OPEN PROTOCOL var_len_file IO_DEVICE "logfile.txt"
MODE write STREAM out_file STATUS status
IF status
    GOTO error
END
FOR ii = 1 TO 10
    IO_WRITE STREAM out_file DATA "Record: ", ii STATUS
    status
    IF status
        GOTO error
    END
END
IO_CLOSE STREAM out_file STATUS status
IF NOT status
    GOTO done
END
error:
MESSAGE status TEXT logfile.txt
done:

```

10.6 Example Serial Driver Sequence

The example below shows how I/O Server statements can be used in an AIM sequence for serial I/O.

```

SET run = -1

I/O_OPEN PROTOCOL simple_serial IO_DEVICE "serial:1>"
MODE read_write STREAM io STATUS status
IF status
    GOTO error
END

WHILE run
    IO_READ STREAM io DATA input STATUS status
    IF status
        GOTO error
    END
    SETS command = input LEFT_OF " ,"
    IF command == "MOVE"
        CALL move_cmd
        cmd_status = 0
    ELSE_IF command == "START"
        CALL start_cmd
        cmd_status = 0
    ELSE_IF command == "STOP"
        CALL stop_cmd
        cmd_status = 0
    END
END

```

```
        ELSE
            cmd_status = -1
        END

        IO_WRITE STREAM io DATA "Status " + cmd_status STATUS status
        IF status
            GOTO error
        END
    END
    END
    GOTO done
error:
    MESSAGE status TEXT serial I/O
done:
    IF io
        IO_CLOSE io
    END
```

Force Sensing Module

11

11.1 Introduction	120
AdeptForce versus Digital Force Sensors	120
Enabling Force Sensing in AIM	120
Configuring V+ for Digital Force Sensors	120
11.2 Modes of Operation	121
Guarded Mode	121
Protect Mode	123
11.3 Operator Interface (AdeptForce Only)	124
Force Sensor Status and Control Display	124
Force Signature Display Page	125
11.4 Force Process Database (AdeptForce and Digital Sensor)	126
Force Process Records	127
Force Process Routines	131
11.5 Example Application	132
Part Acquisition	133
Part Insertion	134

11.1 Introduction

The Force Sensing Module allows you to perform and monitor force-sensing operations using either an AdeptForce sensor or a touch sensor connected to digital I/O. Throughout this chapter, those items that apply only to AdeptForce sensors or digital touch sensors are marked as such.

NOTE: If you are using AdeptForce, refer to the *AdeptForce VME User's Guide* for additional information.

The basic components of the Force Sensing Module are:

- Status and Control displays (AdeptForce only).
Two menu displays allow you to interact with an AdeptForce sensor. One menu shows live force sensor readings and gives the user various force system control capabilities. The second menu allows you to configure and display force signature plots of a force operation, including the ability to zoom in or out.
- Force Process database.
The Force Process database contains the information for performing a complex series of force-guided motions during motion. Process routines specified within the Force Process database are called upon to perform the actual operations.
- Force strategies.
Many assembly strategies can be built from the core force process routines to perform such operations as insertion searches, assembly verification, and snap-fit. These strategies are accessible when using standard AIM statements.

AdeptForce versus Digital Force Sensors

One question that often arises is whether to use an AdeptForce force sensor or a simple digital I/O sensor to perform force sensing. Digital sensors offer lower cost and greater simplicity of use, whereas AdeptForce offers, in a complete package, greater flexibility and immediate functionality simply by installing it between the robot and tool flanges.

The cost advantages of digital sensors usually depend on whether the additional design, construction, and debug costs of a tool incorporating such a sensor can be justified. Typically such sensors are set at a single “trigger” load. For applications that require adjustable forces, AdeptForce may be a better choice. When force levels need to be monitored for production or quality reasons, only AdeptForce offers such capability.

Enabling Force Sensing in AIM

To enable force sensing, you need to make a one-time change to the robot initialization database. After starting AIM, choose **Setup ➡ Initialization Data ➡ robini.db** and set the value of force-sensing option enable to on.

Configuring V+ for Digital Force Sensors

To get the fastest possible response to detected contact, the servo boards latch the motor positions at the instant of contact. For AdeptForce sensors, this is configured automatically, but for digital sensors, this requires that the SYS IO module be configured to send an interrupt on the backplane when the signal transitions from low to high and

that the servo board be configured to latch positions upon detecting the backplane interrupt.

To perform this configuration, run CONFIG_C (as described in the *Instructions for Adept Utility Programs*) and edit digital input configuration and servo board configuration as follows:

- Set up digital input 1001, 1002, or 1003 to generate a position latch signal on the backplane.
- Set up every servo board for the desired robot to latch positions based on the backplane latch signal (it must match the position latch used in the previous step).

Refer to section 6.6 in the *Adept MV Controller User's Guide* for more details on configuring digital input signals.

11.2 Modes of Operation

Measuring forces is often of little use in itself: What matters is how the robot system responds to those forces. AdeptForce provides two standard ways of responding to sensed forces. Guarded Mode rapidly halts the robot when a force or moment crosses a specified threshold. Protect Mode is a simple crash-detection scheme that turns off robot power when a specified force level is exceeded. It operates like a panic button that is struck whenever a sufficiently hard crash occurs. A summary of the three operating modes is given in Table 11-1.

NOTE: The digital force sensor operates in Guarded Mode only.

Since each of these modes may influence robot motion, any force system hardware error that occurs while the modes are active will cause the robot to panic stop with a “*Force sensor communication error*”. Once the problem is fixed, the user can enable robot power again and proceed. Protect Mode is then automatically reenabled, but Guarded Mode must be explicitly reenabled after a force system hardware error.

Table 11-1

Summary of Force System Operation Modes

	Guarded Mode	Protect Mode
Cause of trip	Either of two trip conditions: - force or moment - directional, planar, or resultant - trip direction: X, Y, or Z - greater-than/less-than flag	Force or moment resultant exceeds present threshold
Effect of trip	Halt robot exactly at trip position	“Panic stop” stops robot arm
Recovery required	None — ready for next motion	Robot power must be reenabled
State after trip	Guarded Mode disabled	Protect Mode remains enabled

Guarded Mode

When a Guarded Mode trip occurs, the current robot position is automatically recorded and the robot is commanded to return to the trip location as quickly as possible. Some

overshoot will naturally occur, since it is impossible to stop a mechanical system instantaneously. The overshoot is a function of robot speed, load, the direction of the motion, and the “give” of the surfaces in contact. Once the robot returns to the trip location, the next motion instruction can be processed.

Guarded Mode can have up to two specified trip conditions operating concurrently. A force trip occurs when either of the conditions trip. After the motion, the force system reports the condition (if any) that tripped, but the application program must decide how to respond to each of the possible trip conditions. For example, you could consider the first trip condition to signify a “normal” force trip and the second trip condition to signify an error condition.

Guarded Mode is used to implement a *guarded move*. Guarded moves are robot motions that are halted while traveling to a goal location if a force trip occurs. To perform a guarded move, you typically send the force system the information that configures the trip conditions for Guarded Mode, start the move, and then enable Guarded Mode. Guarded Mode remains enabled until a force trip occurs, or until it is explicitly disabled. As a result, if a force trip does not occur during a guarded move and Guarded Mode is not disabled, all subsequent moves automatically become guarded until a trip finally occurs.

There are three type of trip conditions that can cause a Guarded Mode trip: directional (1-dimensional), planar (2-dimensional), and resultant (3-dimensional). Each trip condition actuates on either a force or moment, and each has lower and upper thresholds. If the force or moment level falls below the lower threshold or rises above the upper threshold, the condition trips.

Directional trip conditions measure forces or moments in a single direction (1 dimension). You can select the X, Y, or Z direction as specified by the Force Reference Frame (FRF). Directional trip conditions are particularly useful in applications where it is helpful to ignore forces in certain directions, such as X-Y forces that can occur while decelerating toward a location where only Z direction forces are important to an insertion.

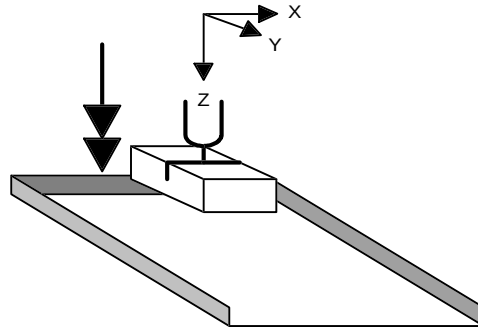
Planar trip conditions monitor forces or moments in a plane (2 dimensions). You can select the X, Y, or Z direction as specified by the Force Reference Frame. Planar trip conditions monitor forces in the plane perpendicular to the specified axis. For example, suppose the robot should stop if it senses more than 1 lb. of lateral force (any force in the X-Y plane). The X-Y plane is selected by specifying the Z-axis (because it is perpendicular to the X-Y plane).

Note that the same direction has very different meanings for directional and planar trip conditions. Directional trip conditions monitor forces *along* the specified axis; planar trip conditions monitor forces in the plane *perpendicular* to the specified axis. The measured force can be negative in the case of the directional trip condition if it is in the opposite direction from the axis. The measured force is never negative for a planar trip condition since it does not distinguish between different directions on the surface of the plane.

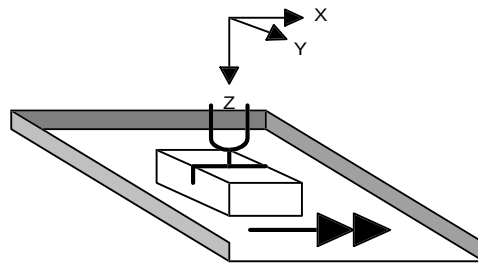
Resultant trip conditions measure net force or moment in all directions (3 dimensions). Since this type of trip condition is nondirectional, no direction needs to be specified. Like planar trip conditions, the measured force can never be negative since this condition measures only the magnitude of the force. Resultant conditions are the easiest to use of the three possibilities, since the user need only specify two thresholds and whether to monitor force or moment.

An in-depth application example should make the concepts clear. Assume that you would like to place a package onto a pallet and secure it against a corner of that pallet. You could perform the operation, shown below, in three stages:

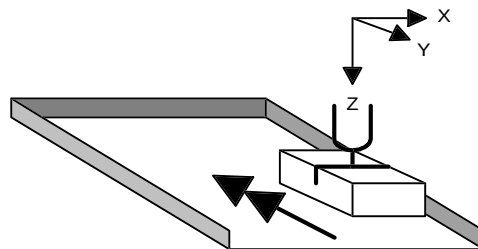
1. Guarded move to the surface. Since you may be approaching the location with high X-Y deceleration, you perform a directional guarded move, monitoring forces in the Z direction only.



2. Guarded move to the side of the pallet. If the point of contact with the surface is more than 5 inches from the center of the force sensor, the sensor is more sensitive to assembly moments than to forces. Since a moment will be created about the Force Reference Frame (FRF) origin, you perform a resultant guarded move, monitoring moments about any axis. (This move also could be performed with a directional guarded move, measuring forces in the FRF X direction or moments about the FRF Y-axis.)



3. Guarded move to the corner of the pallet. Since you already have a moment caused by pressing up against one wall, you perform a directional guarded move, measuring moments about the FRF X-axis. (This move also could be performed with a directional guarded move, measuring forces in the FRF Y direction.)



Protect Mode

Guarded Mode is typically in effect for only a small percentage of the total cycle time for a part acquisition and placement. Although it is unlikely that an unexpected collision will occur during the rest of the cycle, Protect Mode can provide a level of emergency sensing of such forces.

Protect Mode requires specifying a force level that, if exceeded, will apply robot brakes and turn off arm power. This differs from Guarded Mode in that a Guarded Mode trip leaves the robot able to begin its next move immediately. After a Protect Mode trip, however, the user must reenables arm power from either the system terminal or from the manual control pendant. Once power is reenables, robot motion can be continued.

11.3 Operator Interface (AdeptForce Only)

The AdeptForce operator interface for the Force Sensing Module consists of two displays. The Force Sensor Status and Control display allows you to observe the forces exerted by the sensor and to control the force signature buffer. The Force Signature Buffer display allows you to interactively display and modify the force, moment, and position graphs within a window.

Force Sensor Status and Control Display

To access the Force Sensor Status and Control panel:

I/O ➡ Force Sensor Controls

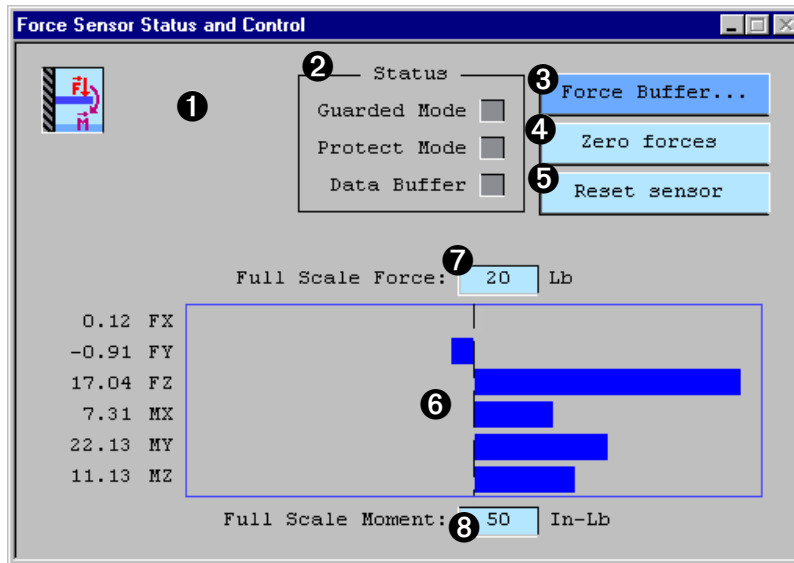


Figure 11-1

Force Sensor Status and Control Menu Page

- ❶ If your system has multiple force sensors, this data box allows you to choose among them. The forces and moments detected by the selected sensor are displayed on the bar charts and collected by the force buffer if it is under manual control.
- ❷ Read-only flags that indicate the status of the Guarded and Protect operational modes, as well as the status of the force signature buffer. The status flags update automatically during force acquisition and insertion operations.
- ❸ Choose **Force Buffer** to move to the force buffer display menu.
- ❹ Choose **Zero forces** to re-zero the force sensor readings.
- ❺ Choose **Reset sensor** to reinitialize the force sensor. Force readings may suddenly change as the force offsets are cleared.
- ❻ The bar chart displays the current force and moment levels.
- ❼ The bar-chart force scaling can be changed by entering a value in this data box.
- ❽ The bar-chart moment scaling can be changed by entering a value in this data box.

Force Signature Display Page

The “Force Signature Display” menu page is accessed by choosing **Force Buffer** from the Force Sensor Status and Control menu page.

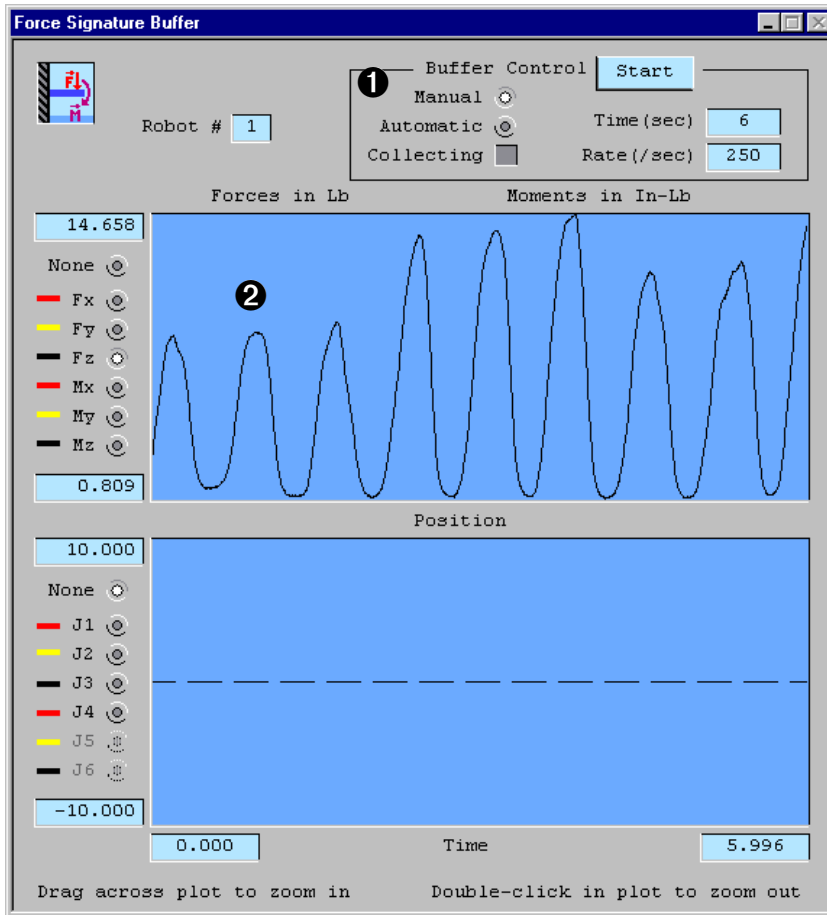


Figure 11-2
Force Signature Display Menu Page

- ① The buffer controls specify how force data is to be collected. You can specify the time between force samples and the total duration of data collection.

Manual	Allows you to activate buffer collection by choosing Start .
Automatic	Automatically activates buffer collection based on the parameters defined on the Force Process menu page. See Figure 11-5 for details.
Collecting	Read-only flag that indicates the status of the data collection process.
Time (sec)	Specify the duration (in seconds) for the data collection process.
Rate (/sec)	Specify the number of samples per second.

- ② The force/moment and position graphs show the most recent data of the type selected by the radio buttons. The limits of the plot are shown in the value boxes to the left of and below the plots.

There are three ways of changing these limits:

- Zoom in using the mouse by pressing the left mouse button and dragging across the region of interest.
- Zoom out using the mouse by double-clicking anywhere in the plot. The plot will be automatically rescaled to fit all the data within its bounds.
- Type the desired scales into the value boxes to the left of and below the plots.

11.4 Force Process Database (AdeptForce and Digital Sensor)

The Force Process database contains sets of records that describe a series of motions to perform during part acquisition or insertion and how to interpret the contact conditions that occur. One or more records comprise a complete process. For example, an insertion process that includes post-insertion verification might consist of two Force Process records, one for insertion search and the other for insertion verification. The series of Force Process records used in a single process is described in an index record like that shown in Figure 11-3.

NOTE: The force sensing process is called during the move from the approach position to the final location.

To edit the Force Process database:

Edit ➡ Force Process

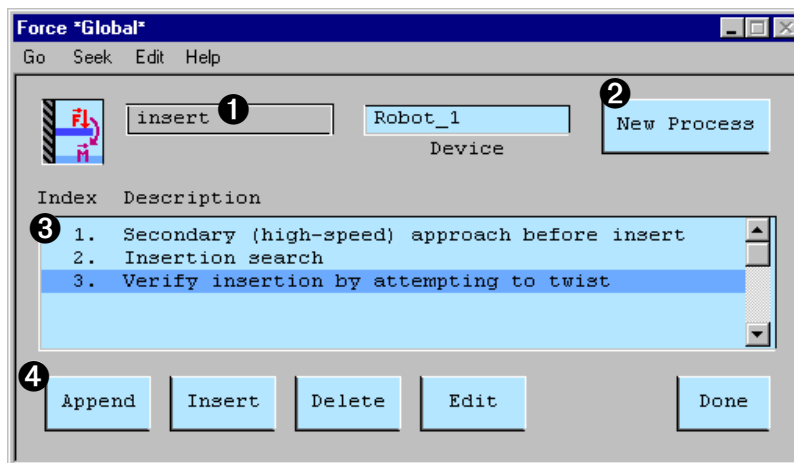


Figure 11-3
Force Process Index Page

- 1** This is the name of the force process that must be entered into the Location database force process data box, as shown in Figure 11-4. See “Location Record Options” on page 74 for details.

NOTE: The location strategy routine *will not* be called when a force process is used.

The screenshot shows a window titled "Location Strategies". It contains four distinct sections, each with a title bar and a set of controls:

- Location:** Features two radio buttons, "V+ Routine" and "AIM sequence". The "AIM sequence" button is selected. To the right is a text input field containing the text "rn.mw.standard".
- Approach:** Features the same two radio buttons, both of which are unselected. To the right is an empty text input field.
- Depart:** Features the same two radio buttons, both of which are unselected. To the right is an empty text input field.
- Force Sensing:** Features a checked checkbox labeled "Enable". To the right is a text input field containing the text "insert".

Figure 11-4
Location Strategies Page

- 2 Choose **New Process** to create a new force process. A pop-up window is displayed requesting a name for the new force process.
- 3 Displays all of the force process steps for this force-guided operation. Click any force process record name to select it. Then, choosing any of the buttons on the bottom of the menu page will affect the currently highlighted force process record.
- 4 Choose **Append** to add a new force process record to the end of the list.
 Choose **Insert** to add a new process above the currently highlighted process.
 Choose **Delete** to delete the highlighted process. If there is only one force process step, this will also delete the force process.
 Choose **Edit** or double-click the desired force process to display the force process editing screen.
 Choose **Exit** or click the close box in the upper left-hand corner to exit.

Force Process Records

Figure 11-5 is a sample force process record. Force processes often require a series of steps. Each record contains the name of a “process routine” to perform the specified operation, as well as the data to be used during the operation. It is likely that the core set of routines provided with the Force Sensing Module will be the only routines necessary for typical applications.

Force Process "Global"

1 of 1

1 insert 2 Robot_1 11-Oct-96 07:21

3 Description: Secondary (high-speed) approach before insert

4 Process routine: rn.fp.standard Learn rate (%): 0

5 Edit Trip Conditions

6

1. Disabled

2. Disabled

7 Search Parameters

8

Relative Position						
X	Y	Z	y	p	r	
0.000	0.000	-2.000	0.000	0.000	0.000	

Line/	Joint	Speed	Rot.	in/	mm/	Coarse	Accel	Decel	Prof
			Speed	sec	sec	I Fine			
		25	0			☒	100	100	0

9 Here 10 Prev Next

Figure 11-5

Force Process Database Record

- 1 Shows the name of the force process currently being edited. The record number and the total number of records for this force process are displayed beneath the name.
- 2 Shows the motion device associated with this force process.
- 3 This is an optional data box used to describe the current force process step. This data box is displayed in the Force Process index page.
- 4 Specify the force process routine in the Process routine data box. The standard Guarded Move routine supplied with the Force Sensing Module is **rn.fp.standard**.



CAUTION: When an error condition occurs, the operator has to make the decision to Proceed or to Abort the operation. This decision must be made based on the location of the robot in the workcell and what the next move is in the Force process. To avoid damage to the robot or tooling, you may want to write a custom routine that *always* returns the robot to a "safe" starting position whenever an error condition occurs. See the *AIM Customizer's Reference Guide* for details on writing and customizing routines.

The routine **rn.fp.search** can be specified if a search should be performed after the Guarded Move fails. If **rn.fp.search** is specified, the **Search Parameters** button is enabled (see item 7). If **rn.fp.search** is specified, you can also specify a rate (%) in the "Learn Rate" data box so that the insertion location will be learned (updated). This feature allows the system to update the location based on the locations found in previous searches. Additionally, it can reduce cycle times due to a contact error in the location or a gradual drift in the physical location or tooling.

Customizers may want to modify these routines. For example, to actuate end effectors or workcell hardware in conjunction with acquisition or insertion, you may want to make a copy of **rn.fp.standard** and add the appropriate program instructions. New routines may be created by copying one of the standard routines and modifying it for particular needs.

- 5 Specify any conditions that you want to initiate a Guarded Move stop. Choose the **Edit** button to change the displayed trip conditions (this option does not apply when using digital force sensors). The following screen is displayed:.

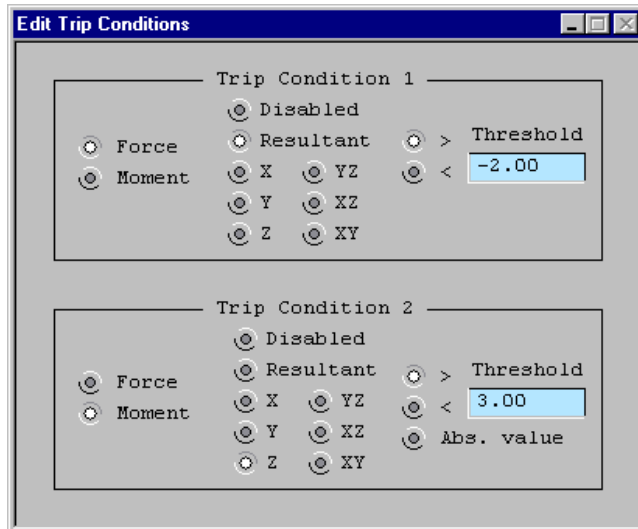


Figure 11-6
Edit Trip Conditions Page

Disabled	Select this radio button to disable the trip condition.
Force/Moment	Specify either force detection or moment detection. See the <i>AdeptForce VME User's Guide</i> for details.
Resultant	Select this radio button to detect a force/moment value from any axis or plane; or, use the radio buttons below Resultant to specify a particular axis or plane.
Threshold	Specify the threshold value in this data box (either negative or positive). The trip condition is initiated for any detected force/moment that is less than or greater than this value by selecting the corresponding radio button.



CAUTION: If the sign of the value is not correct and a contact is sensed, the trip will not occur and damage to the equipment may result.

Absolute Value	Select this radio button to use the absolute value of <i>any</i> detected force/moment (either negative or positive) from a specified X, Y, or Z-axis. This option eliminates the need to determine the correct sign for the Threshold value.
----------------	---

NOTE: Absolute Value is not available for XZ, YZ, XY, or Resultant.

- 6 Select the condition defining a successful force operation whether a trip (contact) is expected or not.

- | | |
|---------------------|--|
| Use digital sensor | Select this checkbox when you are using a digital force sensor. <i>Do not</i> select this checkbox when using AdeptForce. |
| Force trip expected | Select this checkbox if a force trip is expected during the move for this step of the force process. If no force trip occurs, an error condition results. Do not select this checkbox if a force trip is not expected. If a force trip occurs, an error condition results. |
| Zero forces before | Select this checkbox to zero all forces prior to initiating the move for this step of the force process. |
| Start buffer before | Select this checkbox to start the data collection buffer prior to initiating the move for this step of the force process. |
| Stop buffer after | Select this checkbox to stop the data collection buffer after completing the move for this step of the force process. |
- 7 Decide whether you want to perform a search if the operation fails. This is normally appropriate only for insertion operations. See Figure 11-11 for details.
 - 8 Specify the robot motion goal location. If this is the first record in the force process, it is interpreted as an offset to the nominal goal location (obtained from the location record). In subsequent records, this data box represents an offset relative to the point of completion of the previous process.
 - 9 Choose **Here** to record the goal location as the current robot position. For force process applications, locations are defined as offsets relative to locations determined during sequence execution. As a result, these offsets can be taught only during walk-thru training.
 - 10 These buttons allow you to move between force process records.

Force Process Routines

The standard force process routines are listed in Table 11-2. You need to specify one of these routines in the Force Process database “Process routine” data box.

Table 11-2
Standard Force Process Routines

Routine Name	Function
rn.fp.standard	Moves the robot toward a goal location, monitoring for a force trip. If a force trip occurs and the “trip expected” flag is set, or does NOT occur and is NOT expected, the operation is considered a success. Otherwise, the operation fails and the error ‘rn.opr.fail’ is returned.
rn.fp.search	Moves the robot toward a goal location. If a force trip occurs and the “trip expected” flag is set, or does NOT occur and is NOT expected, the operation is considered a success. Otherwise, a “square spiral” search is performed, attempting a series of moves. If any of those moves completes successfully, the search terminates. If successful, the insertion location will be learned at a rate set by the “Learn Rate” data box. (If the learn rate is 0%, the database is never changed; if the learn rate is 100%, the location in the database is always replaced by the actual location. For learn rates between these values, the location written to the database is a weighted average of the two locations.) If each step of the search fails, then the error rn.opr.fail is returned.

11.5 Example Application

As an example of the concepts presented in this chapter, consider the classic peg-in-hole insertion task.

Typical process steps for an acquisition and peg-in-hole insertion are:

1. Acquire the peg from a feeder.
Using the force sensor to determine if a part is in the feeder and register the gripper against the top of the part, the robot will acquire the peg from a feeder in a known location.
2. Approach the goal location.
The robot will use a two-stage approach. First, it will perform a standard high-speed approach (25 mm above the goal location) to assure that the robot tool moves to the assembly at a safe height. Next, it will perform a much closer approach to the location. Since this secondary approach is to a point above the surface of the assembly, surface contact is not expected, so this motion also can be very fast.

Using a secondary approach point can sometimes improve cycle time, since it involves a fast close approach before entering the slower guarded move to the assembly location rather than going directly into the guarded move from the original approach location.

3. Insert the part.
The part will be inserted using the standard spiral search pattern, if necessary. Since possible misinsertion may require a quick stop, this motion is slower than the approach moves. Because it is a short motion due to the secondary approach, it has little effect on cycle time.
4. Verify the insertion.
Confirm that the peg is properly inserted by attempting to move to one side. The force sensor will detect if the peg contacts the edge of the hole.

The AIM statements for inserting the peg might be as follows:

```
MOVE TO feeder  
MOVE TO hole
```

The force processes used to acquire a part from a feeder and insert it at a location will be described in the subsequent sections. In order to demonstrate stop-on-force using both AdeptForce and digital force sensors, the acquisition will be performed with a digital force sensor and the insertion with an AdeptForce sensor.

Part Acquisition

Figure 11-7 shows the “strategy” page for the feeder location. To use a force process, you must select the Force Sensing Enable check box and choose a force process as shown in the figure.

NOTE: The Location record’s main menu page indicates that the Force Sensing option is enabled. This allows you to easily identify those records that use Force Sensing.

The screenshot shows a window titled "Location Strategies". It contains four distinct sections, each with a title bar and a set of controls:

- Location:** Features two radio buttons, "V+ Routine" and "AIM sequence". The "AIM sequence" radio button is selected. To the right is a text field containing the value "rn.mw.standard".
- Approach:** Features two radio buttons, "V+ Routine" and "AIM sequence". The "V+ Routine" radio button is selected. To the right is an empty text field.
- Depart:** Features two radio buttons, "V+ Routine" and "AIM sequence". The "V+ Routine" radio button is selected. To the right is an empty text field.
- Force Sensing:** Features a checked checkbox labeled "Enable". To the right is a text field containing the value "acquire".

Figure 11-7
“Strategy” Page for Peg Feeder Location

To pick up the peg, the gripper will first move until contact is detected. If no contact is sensed, an error will be generated. After contacting the part, the robot will then close the gripper and depart. The Force Process record, “acquire” (shown in Figure 11-8), contains the additional information required to perform the force-guided action.

Force Process (mowsam)

1 **acquire** Robot_1 11-Oct-96 07:31

2 1 of 1

Description: Touch part before acquire

3 Process routine: rn.fp.standard Learn rate (%): 0

4 ☒ Use digital sensor
☒ Force trip expected

Search Parameters

Relative Position						
X	Y	Z	y	p	r	
0.000	0.000	5.000	0.000	0.000	0.000	

6

Line/Joint	Speed	Rot. Speed	in/sec	mm/sec	Coarse	Fine	Accel	Decel	Prof
☐	10	0	☐	☐	☐	☒	100	100	0

Here Prev Next

Figure 11-8

Part Acquisition Example, Force Process Record 1 of 1

- 1 Name. The name of the Force Process. The process must be specified in Location database "Force process" data box for part acquisition. In this case, the name is "acquire".
- 2 Record index. Since the force process routine **rn.fp.standard** performs the required guarded move to an object, only one record is required for the acquisition process. This record is, therefore, record "1 of 1".
- 3 Process routine. The routine **rn.fp.standard** is specified, since it is desired to move until a force trip is detected.
- 4 Trip conditions. The Use digital sensor and Force trip expected check boxes are checked to indicate that success is achieved when the digital sensor detects part contact.
- 5 Relative position. The robot will proceed 5 mm beyond the nominal Location database position during the guarded move.
- 6 Speed. The "speed" data box scales the guarded move speed. It overrides any speed setting associated with the location in the Location database.

Part Insertion

Since the desired insertion strategy is to move at high speed to a secondary approach location, insert the part and then verify the insertion (three Force Process records are required). The force process routines **rn.fp.standard** and **rn.fp.search** perform the functions of the secondary approach (guarded move with no force trip expected), the insertion search, and the insertion verification (guarded move with a force trip expected).

The first Force-Process record might appear as shown in the following figure.

The screenshot shows the 'Force Process *Global*' dialog box. It contains the following fields and controls:

- 1** Name:
- 2** Record index:
- 3** Description:
- 4** Process routine: Learn rate (%):
- 4** Trip Conditions:
 - 1. Disabled
 - 2. Disabled
-
- 5** Relative Position:

X	Y	Z	V	p	r
0.000	0.000	-2.000	0.000	0.000	0.000
- 6** Speed settings:

Line/Joint	Speed	Rot. Speed	in/sec	mm/sec	Coarse	Fine	Accel	Decel	Prof
☐	25	0	☐	☐	☐	☒	100	100	0
-

Figure 11-9
Part Insertion Example, Force Process Record 1 of 3

- 1** Name. Must be the same name as that specified in Location database “Force process” data box for part insertion. In this case, the name is “insert”.
- 2** Record index. Since this is the first of three Force-Process records used to perform and verify the insertion, it is labeled record “1 of 3”.
- 3** Process routine. The routine **rn.fp.standard** is specified.
- 4** Trip conditions. By not specifying any lower or upper thresholds, the trip conditions will be effectively disabled (the thresholds will be set to extremely large values). The move is, therefore, made without any force detection.
- 5** Relative position. Since it is desired to approach the assembly database location by 2 mm and the positive tool Z direction is toward the surface, the relative location may be set as a -2 mm offset in the Z direction. The relative location can also be taught during walk-thru training.
- 6** Speed. The speed is set to a high value during this motion because surface contact is not a concern. The “Speed” data box overrides any speed setting associated with the location in the Location database.

The second record of the force process governs the part insertion. A typical record to perform this task is shown in the following figure.

Force Process "Global"

insert Robot_1 11-Oct-96 07:39
2 of 3

1 Description: Insertion search

Process routine: rn.fp.search Learn rate (%): 10

2 Edit Trip Conditions

1. Stop-on-force if Fz > 1.00

2. Disabled

3 Search Parameters

4 Relative Position

X	Y	Z	γ	p	r
0.000	0.000	2.000	0.000	0.000	0.000

Line/Joint	Speed	Rot. Speed	in/sec	mm/sec	Coarse Fine	Accel	Decel	Prof
10	0				Coarse Fine	100	100	0

Here Prev Next

Figure 11-10
Part Insertion Example, Force Process Record 2 of 3

- 1 Process routine. The routine **rn.fp.search** is specified to perform an attempted insertion followed, if necessary, by a square spiral search.
- 2 Trip conditions. There are several possible ways of monitoring forces during the insertion attempt. One is to use a directional trip condition (dimension 1), which monitors forces only in the Z direction. The other way (as shown in the figure) would be to monitor the resultant force (dimension 3). Since contact is expected only along the direction of motion, the resultant force and directional force should be approximately the same. The trip threshold is set at 1 lb.
- 3 Search parameters. These are set depending on the assembly tolerances and the positioning repeatability of both the part in the gripper and of the assembly. Typically, these values are empirically determined. In this case, they are set assuming that the part-to-assembly repeatability is better than ± 0.5 mm, and that the maximum part-to-assembly misalignment that still results in a successful insertion is ± 0.25 mm and that there is no rotational misalignment.

	X (mm)	Y (mm)	R (deg)
Step size:	0.25	0.25	0
Max search:	0.5	0.5	0
Depart height:	2		
Depart speed:	50		

Figure 11-11
Search Parameters

The “Depart height” is the distance the gripper will depart from the last trip location before its move to the next approach location during the search. Selection of this value depends on the compliance of the gripper, the part, and the assembly. A small value would cause the component to drag along the surface.

The “Depart speed” governs the speed of the motion away from the assembly after a failed insertion attempt and the speed of the motion approaching the next search location. This value is generally several times larger than the value in the Force process “Speed” data box, which governs the force-guided move to the surface itself.

- ④ Relative position. The previous motion left the robot 2 mm above the goal location. The relative location for this move must therefore be 2 mm in the positive Z direction. Speed determines the speed of the move and overrides any speed setting associated with the location in the Assembly database. A low speed setting for the move to the surface is necessary for delicate components.

To maximize this speed with acceptable force overloads during contact, examine the forces during assembly from the “Force sensor status” menu page or display a force signature after an assembly.

The final record verifies the insertion by performing a rotational guarded move to check that the component strikes the side of the hole (Figure 11-12).

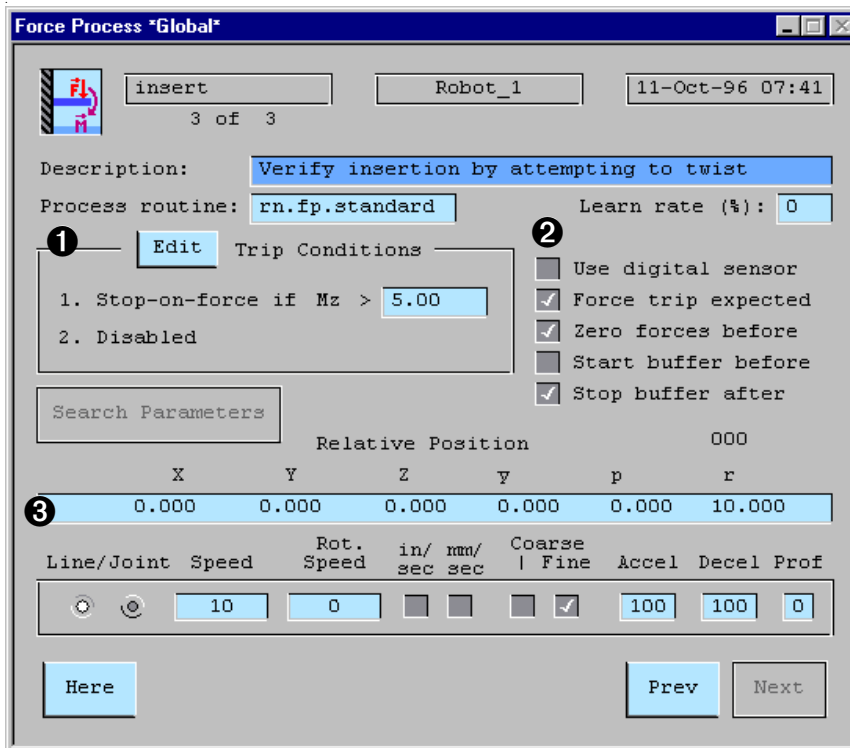


Figure 11-12

Part Insertion Example, Force Process Record 3 of 3

- ❶ Trip conditions. Since we are rotating about the Z-axis in a positive direction, we expect a positive moment about the Z-axis if contact is detected.
- ❷ Offset. We want to zero forces before the motion to ignore any forces that may have been incurred up to that point.
- ❸ Relative position. The previous motion left the robot at the insertion location. To confirm the assembly of a square peg into the hole, we try to rotate about the Z-axis. If contact is detected, the verification succeeded.

Using Conveyor Systems

12

12.1 Indexing vs. Tracking Conveyors	140
Conveyor Database, Frame Database, and Location Database	140
Conveyor Terminology	140
Conveyor Objects	141
12.2 Conveyor Overview	141
12.3 Indexing Conveyor Records	142
12.4 Using an Indexing Conveyor	143
Indexing Conveyor: Runtime Operation	144
12.5 Conveyors and Pallets	144
12.6 Tracking Conveyor Strategy	144
12.7 Tracking Conveyor Frame Records	146
12.8 Tracking Conveyor Options	147
12.9 Using a Tracking Conveyor	150
Single-Access Operation	151
Multiple-Access Option	151
Variable-Access Option	152
12.10 Calibrating a Tracking Conveyor	152
Belt Calibration Procedure	153
Calibrating Multiple Robots with a Single Conveyor	154
Belt Calibration and External Files	154
12.11 Conveyor Queue Summary	155
12.12 Conveyor Statements	157
GET_FRAME Statement	157
RELEASE_FRAME Statement	160
CHECK_OBJECT Statement	160
MATCH_OBJECTS Statement	161
RESUME_CONVEYOR Statement	161
12.13 Conveyor Tracking Example	162
Create a Resource Module	163
Create a Conveyor Record	163
Create a Frame Record	164
Create a Location Record	165
Calibrate the Tracking Conveyor	165
Install an Object Sensor	166
Create a Sequence	166

12.1 Indexing vs. Tracking Conveyors

MotionWare can be used in conjunction with two types of conveyor systems, indexing and tracking. In an indexing conveyor system (also referred to as a noncontinuous conveyor system), you specify either control signals or a time interval for the belt to move between stops. When the conveyor stops, the robot removes parts from the belt and the conveyor is signaled to move again. The conveyor must be equipped with a device that can use digital output to turn the conveyor on and off (see Chapter 9).

With a nonvision indexing system, the part must be in the same location each time the belt stops. In a vision-equipped indexing system, a fixed-mount camera takes a picture when the belt stops and the robot accesses any objects found.

In a tracking conveyor system, the belt moves continuously and the robot tracks parts until the speed and location of the robot gripper match that of a part on the belt. The robot then accesses the part. Tracking conveyors must be equipped with an encoder (available from Adept).

With a nonvision tracking conveyor, a sensor signals that a part has passed a known location. MotionWare tracks the progress of the belt and accesses the part when it comes into the robot working area (belt window). Parts must always be in the same location with respect to the center line of the belt.

With a vision-equipped tracking conveyor, a fixed-mount camera takes regularly timed or spaced pictures of the belt and returns the location of parts it locates. Digital I/O also can be used to trigger the picture. These part locations are queued and accessed by the robot. The vision system allows detection of parts which are randomly positioned and oriented on the belt.

Conveyor Database, Frame Database, and Location Database

In MotionWare, conveyor records are associated with records in the frame database. For tracking conveyors, this creates “moving” frames that are constantly updated as the conveyor moves, thus requiring conveyor belts to be calibrated.

Belt calibration, a simple procedure performed from within MotionWare, relates belt direction of travel to the robot work space. Calibration establishes the relationship between an encoder tick and the distance moved by the belt. Once a belt is calibrated and associated with a conveyor record, it can be used by the reference frame for location records.

The following sections discuss creating and calibrating conveyors and using nonvision equipped conveyors. Chapter 15 discusses using the vision option with conveyors.

Conveyor Terminology

Figure 12-1 shows a typical robot cell equipped with a conveyor belt. The direction of travel determines the upstream and downstream ends of the conveyor. In addition to the normal limits on the robot range of motion, a belt window must be created for a tracking belt to define the belt area to be accessed by the robot. This window limits robot belt access, and it also is used to determine whether the robot can successfully track and pick up a moving part.

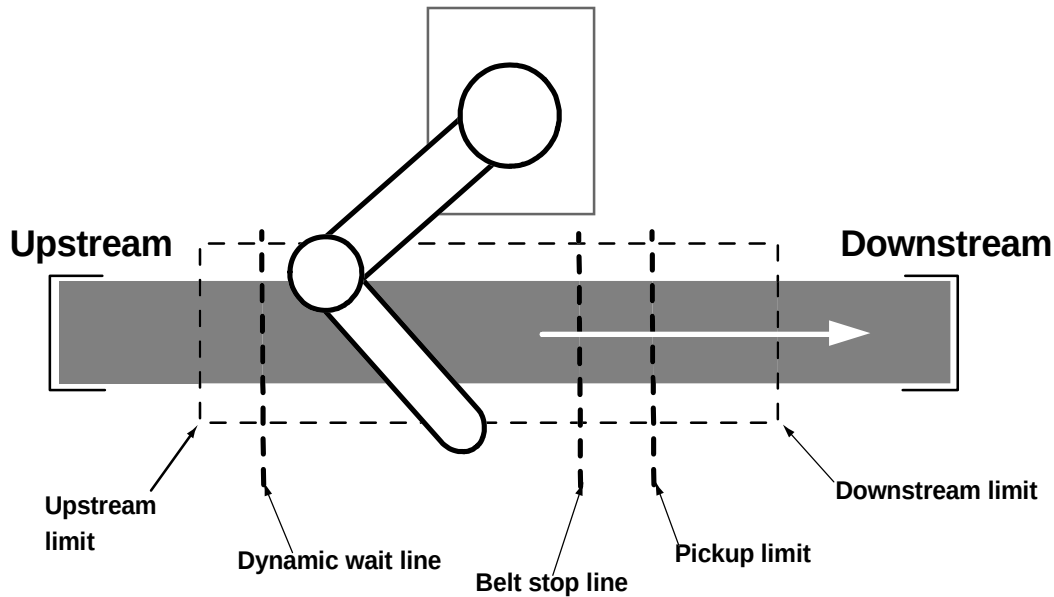


Figure 12-1
Conveyor Terminology

Conveyor Objects

Many operations can be performed on objects placed on a belt, e.g., parts can be acquired, pallets can be filled, assemblies can be built, and objects can be inspected. In this manual we refer to an object that is located and tracked as the “conveyor object”, regardless of what actions are performed on it.

12.2 Conveyor Overview

The MotionWare conveyor option provides a way of creating sophisticated robot/conveyor implementations. These implementations, which include sorting operations, range from pick/place conveyor operations to performing multiple operations on the same part. Depending on your requirements, you will set up the conveyor records and statements differently.

There are four primary classes of conveyor operations:

1. Indexing conveyor operations. A conveyor indexes into position, stops, and then waits until objects on the conveyor¹ have been accessed before indexing again.
2. Simple operations that require accessing each object on a moving conveyor only once. The object can be a part that must be picked up, a bucket that must be filled with a part, or an assembly that requires a part to be inserted. The object can be sensed either with a digital sensor or the vision system. This option is referred to as the Single-Access option.
3. More complex operations where multiple operations are to be performed on the same object. For example, the object may be either a pallet upon which several parts will be loaded/unloaded or an assembly requiring parts to be inserted. This option is referred to as the Multiple-Access option.

¹ This conveyor record can be used for indexing tables or any other type of feeder that indexes and then stops while the robot accesses it.

- Operations in which you decide which objects should be accessed with the robot.
Referred to as the Variable-Access method, this also allows you to pass objects on the same belt to a second robot in a dual MV robot system.

After describing the conveyor record, the method for setting up each type of operation will be discussed.

12.3 Indexing Conveyor Records

To create an indexing conveyor record:

Edit ➡ Conveyor ➡ Edit ➡ New Record

The following screen is displayed:

Conveyor (mowsam)

Go Seek Edit Help

sample.index 09-Oct-96 09:31
2 of 2 Robot_1

Description
This is a sample indexing conveyor.

Wait Path: wait.path

Continuous _____ Object Source _____
☒ Yes ☐ No ☐ Sensor ☒ Vision ☐ External

Null or Indexing Conveyor

Define

X/y	Y/p	Z/r
102.240	56.770	45.678
0.000	0.000	0.000

Static Frame

Advance time: 0.00 sec

Delay after stop: 0.00 sec

Robot clearance: 0.00 mm Here

Control Signals

Inputs: Enabled, Ready, Sensor

Outputs: Advance

Define

Figure 12-2
Indexing Conveyor

12.4 Using an Indexing Conveyor

Refer to Figure 12-2.

1. Create a new conveyor record and enter a name and description for the conveyor record.
2. Select a robot or motion device to be associated with this conveyor record, item ❶. All frame and location records that use this conveyor must use the same robot or motion device.
3. Specify a wait path. The robot will move to this path when there are no objects in the conveyor queue and the conveyor must index, item ❷. This path may be ignored by the conveyor statements (see “GET_FRAME Statement” on page 157).
4. Select ☒ **No** for the Continuous option (the continuous option creates a tracking conveyor, the non-continuous option creates an indexing conveyor).
Select ☒ **Sensor** for the Object Source, item ❸.
5. Define the conveyor frame. This frame is a simple static frame that defines the workspace available when the conveyor stops. Choose **Define** to teach a frame. See “Creating Reference Frames” on page 100, item ❹.
6. In the Advance time data box, item ❺, enter the minimum conveyor run time between each empty conveyor queue. In the Delay after Stop data box, item ❻, enter the robot wait time, i.e., when the conveyor stops and becomes ready (prior to conveyor access). This option is used to guarantee that the belt has completely stopped.
7. In the Robot Clearance data box, item ❼, enter the minimum distance that the robot tool tip must be above the conveyor before the conveyor will advance. Ensure that the last location a robot moves to before the conveyor needs to index leaves the robot tool tip higher than this value or the robot will never access the conveyor.
8. Choose **Define**, item ❽, to enter the digital I/O signals used by this conveyor:

Signal Name	Use
Enabled	A soft or input signal that indicates that the conveyor may be used. This signal is not required.
Ready	A soft or input signal that indicates that the conveyor is ready to be accessed by the robot. If the signal number is 0, the conveyor is always assumed to be ready. If the signal is used, it is checked after Advance time has elapsed.
Sensor	If a sensor is defined, the robot waits to access the conveyor until Advance time is complete and the sensor signal is on.
Advance	An output signal that advances the conveyor. When a statement using this conveyor executes, the advance signal is set to the indicated state for a minimum interval equal to Advance time .

9. Before beginning execution of a sequence using this conveyor, turn the conveyor off and (if a soft signal is used) enable the conveyor.
10. Create a frame record that references this conveyor record.
Select the ☒ **Object Definition** and ☒ **NonVision** options.

The Object Enable signal controls whether or not the conveyor manager task places items into the object queue. If this signal is specified and has a FALSE value, the corresponding object is not placed into the queue. Objects already in the queue are not affected.

This signal may be used with sensor-based conveyors to determine an object's identity. The Object Enable signal for the identified object (and no others) should be asserted prior to asserting the Sensor Strobe signal.

11. Make all your conveyor location records relative to the frame created in step 10.

Indexing Conveyor: Runtime Operation

The following runtime algorithm describes the series of steps taken when an executing sequence accesses an indexing conveyor. (Additional operations may be performed depending on which statement references an indexing conveyor location.)

1. Wait until the robot is above the **Robot clearance** limit.
2. Turn on the Advance signal for a minimum of **Advance time**. If **Advance time** is 0, advance until the Ready signal turns off.
3. If **Advance time** is exceeded and the Ready signal turns on, turn off the Advance signal. Wait for the time specified by **Delay after stop**.
4. If defined, wait until the Sensor signal turns off. This signal allows you to perform other operations on the conveyor before the robot begins its access. Scan any enable signals specified in the object frame record to determine which object was detected.
5. Access objects until the queue is empty.
6. Loop back to step 1 (of this procedure).

12.5 Conveyors and Pallets

MotionWare can access pallets on both indexing and tracking conveyor belts. Since a pallet on a tracking conveyor belt must be completely loaded or unloaded during the time the pallet is within the belt window, either the pallets on tracking conveyors must be small or the belt must move slowly.¹ For indexing conveyors, the belt will stop as long as necessary to load or unload the pallet.

To create a conveyor-based pallet:

- For indexing conveyors, press the **Define** button.
- For nonvision tracking conveyors, press the **Sensor Frame** button.

Place a pallet on the conveyor belt and use the three-point method to create an ordinary reference frame for the pallet (see Chapter 8).

Create a location record, select the ☒ **Pallet** option, and complete the Pallet Parameters group as described in "Using Pallets" on page 80. When the location value is first taught during walk-thru training, record the row-1, column-1, layer-1 location as the location value.

12.6 Tracking Conveyor Strategy

In a tracking conveyor, objects must first pass an upstream sensor. When an object is detected by the sensor, it is placed in the *object queue*. The object queue allows MotionWare

¹ In a special case, you can have tracking conveyor belts stop if objects will move out of the belt window before being accessed.

to keep track of multiple objects on the conveyor belt. As objects come into the belt window they are tracked and accessed.

When a sequence using a tracking conveyor is run, more objects may pass the robot than the robot can access.

The following items affect the robot's ability to access objects:

- The size of the belt window
- The speed of the robot
- The speed of the belt
- The pickup and stop limits

As objects pass the sensor or are located by vision (if that option is used), each object is marked as "new" and placed in a queue. When an object is detected, the "Object Enable Signal" specified in the object definition record is tested. If this signal is TRUE, the object is placed in a queue. Otherwise, it is ignored.

The Object Enable signal also can be used to sort or identify objects on a sensor-based conveyor. Each object should have a separate Object Enable signal defined. External sensors then identify the object and assert the Object Enable signal for the object that is found. The Object Enable signal must be valid while the Sensor Strobe signal is asserted. When the Sensor Strobe signal is asserted, the conveyor manager scans all the object definitions associated with the conveyor and queues the one that is enabled.

The object remains in the queue until it passes the downstream belt window limit.

Each object has a state, which changes as the object moves along the belt and is accessed by the robot.

The possible states are as follows:

New	The object has just been placed in the queue and has not been accessed by the robot.
Busy	The object is currently assigned to a robot.
Done	The object was assigned to a robot, but is now released.
Ignored	The object is being ignored by the current robot.
Lost	The object went out of the belt window while it was still assigned to a robot. This state indicates an error has occurred.

A robot task may request a new conveyor object in two ways:

- It may execute a MOVE statement, which contains a location that specifies the object as its reference frame.
- It may execute a GET_FRAME statement.

The robot task then scans the queue for objects using the following criteria:

- The object must be "New".
- The object code must match the code in the GET_FRAME statement.
- The object name must match.
- The object must be upstream of the pickup limit on the belt.

The object that will be selected satisfies all of these criteria and is the farthest downstream.

If no matching object is downstream of the upstream limit, the robot optionally moves to a location near that limit so it is ready for the object when it comes into reach. This location is specified in the WAIT_AT clause of the MOVE statement.

While an object is in the queue, MotionWare monitors its progress along the belt. When the object passes the belt stop limit, the belt advance signal is disabled. This stops the belt

if appropriate hardware is present. The belt is always stopped for busy objects and may be optionally stopped for new objects. It is not stopped for done or ignored objects. Once the belt stops, it does not restart until all of the objects that caused it to stop have become done or ignored and the robot moves above the belt by the amount entered in the **robot clearance** specification.

When an object passes the downstream limit, MotionWare removes it from the queue. What happens to the objects next depends on the state of the object and the export specifications.

- If the object is busy, it is marked as lost and placed in the lost queue. If the robot is tracking the object at this time, a belt window violation will occur.
- If the object is in any other state, it is either discarded or moved to the export queue, depending on the export specifications for the conveyor.

12.7 Tracking Conveyor Frame Records

To create a tracking frame record:

Edit ➡ Conveyor ➡ Edit ➡ New Record

The following screen is displayed:

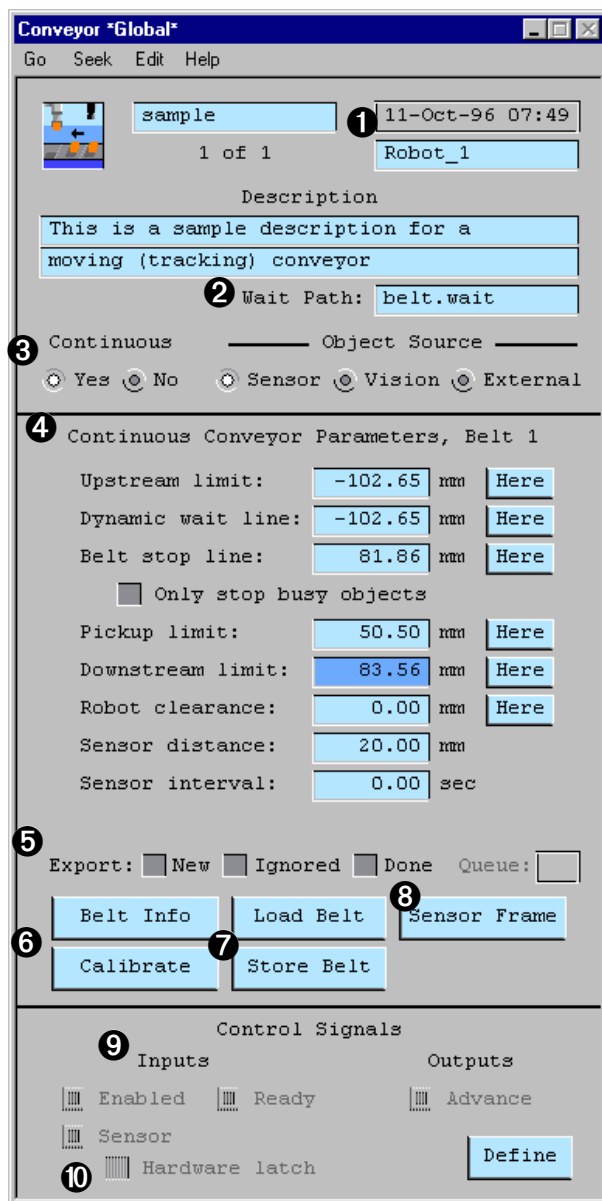


Figure 12-3
Tracking Conveyor

12.8 Tracking Conveyor Options

Refer to Figure 12-3

- ❶ Select a robot or motion device to use with this conveyor record. Double-click this area to display a list of robots and devices.
- ❷ Specify a wait path (optional) that will be used when there are no objects in the queue.
- ❸ Select ☒ **Yes** to make this a tracking conveyor record.

Select ☐ **No** to make this an indexing conveyor record. See “Indexing Conveyor Records” on page 142.

Select the type of sensor used to locate parts of the belt. This chapter will discuss the ☒ **Sensor** option.

- ④ This area defines how the robot will work with the conveyor belt:

Parameter	Use
Upstream limit	Specifies the point on a conveyor belt that must be passed before a robot will begin tracking a part.
Dynamic wait line	In conjunction with the conveyor statements, specifies where the robot will wait if there are objects in the queue, but not yet within the belt window.
Belt stop line	Turns off the belt Advance signal when an object reaches this point. This allows you to stop a belt before objects leave the belt window. (The belt must use the Advance digital signal.) If ☒ Only stop busy objects is selected, the belt will stop for objects that are marked as “busy”. Otherwise, both new and busy objects stop the belt.
Pickup limit	Specifies the point farthest downstream where the robot can begin accessing an object. It helps control belt-window violations by ensuring that the robot has sufficient time to track and access an object before it reaches the downstream limit.
Downstream limit	Specifies the farthest downstream point that the robot is allowed to reach. If the robot is tracking a part when it reaches this point, a belt-window violation will be issued.
Robot clearance	Specifies the minimum vertical distance the robot must be above the belt before the belt can be restarted if it is stopped.
Sensor distance	Specifies the minimum horizontal distance that the belt must move before the next sensor trigger is accepted.
Sensor interval	Specifies the minimum time the belt must be moving before the next sensor trigger is accepted.

- ⑤ This option is used with multiple robot systems. It allows you to specify which objects are passed to a cross-calibrated robot queue.

☒ **New** exports all objects whose status is new.

☒ **Ignored** exports all objects whose status is ignored.

☒ **Done** exports all objects whose status is done.

If any of the export check boxes are checked, the Queue data box becomes active. Enter the number of the export queue to which objects are exported. This queue number

must also be used as the “External input queue” number for an external input conveyor.

- ⑥ Choose **Belt Info** to show the calibration parameters.

Choose **Calibrate** to begin the belt-calibration procedure (see “Calibrating a Tracking Conveyor” on page 152).

- ⑦ MotionWare can use belt-calibration files created with the Belt Cal utility program and/or calibration files that have been exported from other MotionWare systems. Choose **Load Belt** to load a file.

Choose **Store Belt** to create a file that can be used by the belt-calibration utility or another MotionWare system. See “Belt Calibration and External Files” on page 154.

- ⑧ Define the frame location to be queued whenever the sensor signal is asserted.

Choose **Sensor Frame** and you will be able to teach a frame as was previously done for simple reference frames. See “Creating Reference Frames” on page 100.

The Sensor Frame is used to tell the robot exactly where the object is on the conveyor when the sensor is tripped. MotionWare uses the conveyor encoder pulses to track the position of the object on the belt and determine when it reaches the belt window. This feature allows you to use a very long conveyor with the object sensor located outside the robot’s work envelope.

The screenshot shows a software window titled "Define Conveyor Frame". It contains four distinct sections, each with a table of parameters (X, Y, Z, y, p, r) and two buttons labeled "Here" and "Teach".

X	Y	Z	y	p	r
25.000	1160.000	0.000	0.000	0.000	-90.000

Frame Value Compute Here Teach

X	Y	Z	y	p	r
25.000	1160.000	0.000	0.000	0.000	0.000

Frame Origin Here Teach

X	Y	Z	y	p	r
25.000	160.000	0.000	0.000	0.000	0.000

Point on +X axis Here Teach

X	Y	Z	y	p	r
50.000	160.000	0.000	0.000	0.000	0.000

Point near +Y axis Here Teach

Figure 12-4
Define Conveyor Frame Record

- If the sensor is located within the robot’s work envelope, the Sensor Frame can be defined as a standard frame. In this case, simply center the robot pointer over the centroid of the object and use the Teach function to input the frame information.
- If the sensor is located outside the robot’s work envelope, the Sensor Frame must be defined so that it describes the offset between the sensor location (where the object is when the sensor is tripped) and where the object will be when it enters the belt window, as follows:

Point on +X axis - Define a point on the +X-axis of the conveyor that is downstream from the part sensor and inside the upstream limit of the belt window. The +X-axis is parallel to the long axis of the conveyor.

Frame Origin - Define a point on the belt where the sensor is located. This is done by measuring the distance from the "Point on +X axis" to the sensor location.

Point near +Y axis - Define a point anywhere on the +Y side of the conveyor. The orientation of the Sensor Frame (with respect to the World Coordinate system) is computed based on this value.

Frame Value - Choose **Compute** to compute the frame value.

- ⑨ Choose **Define** to specify the following digital signals to be used with this conveyor:

Signal	Use
Enabled	Indicates that this conveyor frame is enabled and can be used.
Ready	Indicates that the belt is ready for access (used by indexing conveyors).
Sensor	Indicates which signal is attached to the sensor that detects upstream parts on a tracking conveyor. This may be left undefined for vision-based conveyors.
Advance	Stops the belt when a tracking conveyor has reached the stop limit.

- ⑩ The ☒ **Hardware latch** is used in conjunction with the high-speed position latch. This latch must be set up using the CONFIG_C utility program. See the *Instructions for Adept Utility Programs* for details. Use of this latch shortens the latency between the time the sensor detects a part and the time the system records the belt position.

12.9 Using a Tracking Conveyor

Create the Conveyor record (refer to the item numbers in Figure 12-3):

1. Create a new conveyor record and enter a name for the record and a description of the conveyor.
2. Select the device associated with this conveyor, item ①.
3. Select ☒ **Yes** for the Continuous option, item ③.
4. Specify ☒ **Sensor** as the Object source.
5. Choose **Define**, item ⑨, to enter any digital I/O signals used by this conveyor. Enter the sensor signal name or number in the Sensor option of the Control Signals, item ⑨.
6. Select any export options, item ⑤.
7. Calibrate the conveyor, item ⑥. This procedure is detailed in section 12.10. Turn off the conveyor when calibration is complete.
8. Define the continuous conveyor parameters. All parameters with a **Here** button can be defined by placing the robot in the appropriate location and then choosing the corresponding **Here** button.

Create the Frame record (refer to “Creating Reference Frames” on page 100):

1. Create a new frame record.
2. Select ☒ **Object Definition** as the frame type.
3. Select ☒ **Non Vision** as the object definition.
4. Double click the Conveyor data box and retrieve the conveyor record created in the above steps as the conveyor record.
5. If you want access to this object to be based on the Object Enable signal, select ☒ **Use object enable signal**.

Create new location records (refer to “Editing Locations” on page 74):

1. Create the required location records and make them relative to the Frame record created above. (Select ☒ **Named** on the location record to display the data box requiring a frame record from the frame database.)
2. Leave the location value blank. Since tracking conveyor reference frames move, this value can be taught only during walk-thru training.
3. Select one of the access options below and complete the steps under that option.

Single-Access Operation

1. Create a sequence that includes a MOVE statement and references locations relative to the conveyor-based frame record.
2. Display the Task Control Panel and start the conveyor server task.
3. Highlight the robot task associated with the conveyor locations. Select the sequence you created in step 1.

Select the ☒ **Walk Thru** option.

4. Place a sample object on the conveyor belt and position it upstream from the sensor. This object must be in the proper position relative to the center line of the belt. (A nonvision tracking conveyor can calculate the object’s position along the conveyor’s direction of travel. However, it always expects the object to be in the same place relative to the center line of the conveyor.)
5. Choose the **Start** button on the Task Control Panel; then choose the **Proceed** push button. Run the conveyor until an object passes the sensor and enters the belt-window area; then turn the conveyor off.

Choose the **Proceed** push button until you get an Undefined location message for the conveyor-belt location record.

Choose the **Edit** button.

6. When the location record is displayed, use the manual control pendant to teach the object’s location. Choose **Teach** (or place the tool tip correctly at the object and choose **Here**). Be careful not to move the object.
7. Proceed with the rest of the sequence, teaching any other undefined values.

Multiple-Access Option

1. Create a sequence that includes MOVE statements referencing locations relative to the conveyor-based frame record. Create a MOVE statement for each access to the conveyor object.

The ☒ **Keep if success** option must be selected.

The ☒ **Keep if error** option must be selected if you want to continue accessing an object even if the previous access failed.

2. Before the MOVE statements, add a GET_FRAME statement. The first argument in the GET_FRAME statement should use the same frame record as the locations in the MOVE statements.
3. After the MOVE statements, add a RELEASE_FRAME statement. The first argument in the RELEASE_FRAME statement should use the same frame record as the locations in the MOVE statements.
4. Display the Task Control Panel and start the conveyor server task.
5. Select the robot task associated with the conveyor locations. Select the sequence you created in step 1.

Select the ☒ **Walk Thru** option.

6. Place a sample object on the conveyor belt and position it upstream from the sensor. This object must be in the proper position relative to the center line of the belt. (A nonvision tracking conveyor can calculate the object's position along the conveyor's direction of travel. However, it always expects the object to be in the same place relative to the center line of the conveyor.)
7. Choose the **Start** button on the Task Control Panel; then choose the **Proceed** button. Run the conveyor until an object passes the sensor and enters the belt-window area. Turn the conveyor off.

Choose the **Proceed** button until you get an Undefined location message for the conveyor-belt location record.

Choose the **Edit** button.

8. When the location record is displayed, use the manual control pendant to teach the object's location. Choose **Teach** (or place the tool tip correctly at the object and choose **Here**). Be careful not to move the object.
9. Proceed with the rest of the sequence, teaching any other undefined values.

The above procedure defines the most basic operation of the multiple-access option. See the description of the RELEASE_FRAME and GET_FRAME statements for other options.

Variable-Access Option

This option is used with vision-based conveyors and is described in Chapter 15.

12.10 Calibrating a Tracking Conveyor

Calibrating a tracking conveyor establishes the relationship between encoder ticks and belt travel. It also establishes a relationship between the belt direction of travel and the world coordinate system. These relationships are used by MotionWare to update the moving-belt reference frame, which allows the robot to track a moving-part location.



CAUTION: Before beginning belt calibration, ensure that the following items have been completed:

- The robot and conveyor hardware are installed. The belt encoder is installed and connected to the controller. See the *Adept MV Controller User's Guide*.

- The external encoder module is loaded. The encoder is assigned to a motion-board channel (this is done automatically for Adept robots). For Adept motion systems, see the description for CONFIG_C in the *Instructions for Adept Utility Programs*.
- A conveyor record exists. See sections 12.3 and 12.7.

During belt calibration, you will place the robot tool tip at the center of a location on the belt, move the belt, and place the tool tip back at the center of this new belt-position location. We recommend you use a pointing device and calibration target similar to the one used with Adept camera calibration products. This equipment consists of a ball-end pointer attached to the robot tool flange and a round disk with a beveled hole in the middle. The bevel allows the pointer to seat accurately in its center.

Belt Calibration Procedure

1. Create a tool transformation to be used during calibration. The tool transformation should describe the difference between the pointing device and the center of the tool flange. See “Tool Transformations” on page 102.
2. From the conveyor record (Figure 12-3), choose **Calibrate**.
3. A window is displayed prompting you to teach the Belt Nominal location. This location can be anywhere on the belt that is within the belt window. Place the tool tip at this location and choose **Here**.

Choose **Continue** to move to the next step.

4. You will now be prompted to teach a location on the belt that is close to the edge of the belt and opposite the robot. Place the robot tool tip at this location and choose **Here**.

Choose **Continue** to move to the next step.

5. You will now be asked to teach a location on the belt that is as far upstream as the robot can reach, and is on the edge of the belt nearest the robot. After this location has been taught, the calibration disk must remain in the same spot on the belt for the next step. We recommend that you tape the calibration disk to the conveyor belt. Place the robot tool tip at this location and choose **Here**.

Choose **Continue** to move to the next step.

6. Move the robot away from the calibration disk (do not move the disk). Run the belt until the calibration disk is as far downstream as possible, but still within reach by the robot. Place the robot tool tip at this location and choose **Here**.

Choose **Continue** to complete belt calibration.

7. Define the belt window (refer to section 12.1 on page 140):

Place the robot as far upstream as it can go while still being able to cover an area of the belt that might contain objects. Choose the **Here** button next to the Upstream limit data box.

Place the robot at the point as far downstream as you want objects to be allowed to travel before the belt stops. This option is used for belts that have an “Advance” signal and you want to be able to stop the belt to ensure that all objects are serviced. Leave the option blank to disable the feature. Choose the **Here** button next to the Belt stop line data box. You can have the belt stop only for objects that the robot is actively tracking (busy objects) or for all objects that enter the conveyor queue. Select ☒ **Only stop busy objects** to have the belt stop only for objects that the robot is tracking.

Place the robot as far downstream as it is allowed to start tracking parts. This option helps you to avoid belt-window violations by not allowing the robot to service objects that are too near the downstream limit. Choose the **Here** button next to the Pickup limit data box.

Place the robot as far downstream as it can go and still cover the area of the belt that might have parts. Choose the **Here** button next to the Downstream limit data box.

Calibrating Multiple Robots with a Single Conveyor

Some operations may require that a single object on a conveyor be accessed by more than one robot. A special feature is included in MotionWare that allows you to define the relationship between two robots¹ on the same belt and pass objects from the first robot's object queue to the second robot's queue.

To use this capability you must:

1. Create a calibration record for each robot and calibrate each of the robots as described in "Calibrating a Tracking Conveyor" on page 152.
2. **Setup ➡ Cross Calibrate.**
3. Select the two conveyors to cross calibrate.
4. Choose **Cross Calibrate** and follow the instructions displayed in the pop-up window.

Belt Calibration and External Files

The Adept utility BELT_CAL.V2 can be used to calibrate conveyor belts. This utility stores belt data on disk in a standard format. MotionWare can load data from these standard files to tracking frame records. Belt-calibration data generated within MotionWare can also be stored to disk in the standard BELT_CAL.V2 file format.

To load or store belt-calibration data, open the conveyor record and choose either **Store Belt** to store a belt-calibration file or **Load Belt** to load a belt-calibration file.

In the displayed pop-up window, enter the name of the belt-calibration file to load/store, select the save set number, and choose either **Load** or **Store**.

NOTE: The Adept belt-calibration utility program uses the file naming convention BELTxx.DAT, where "xx" is a number from 1 to 99. Other Adept programs, such as the advanced camera-calibration program, expect belt-calibration file names that adhere to this convention.

The belt-calibration utility allows you to store six belt-calibration data sets in one file. The MotionWare calibration load procedure will store only one data set to a single file.

¹ Both robots must be controlled from the same controller.

12.11 Conveyor Queue Summary

The current status of a belt can be seen by:

Show ➡ Conveyor Queues

The following screen is displayed:

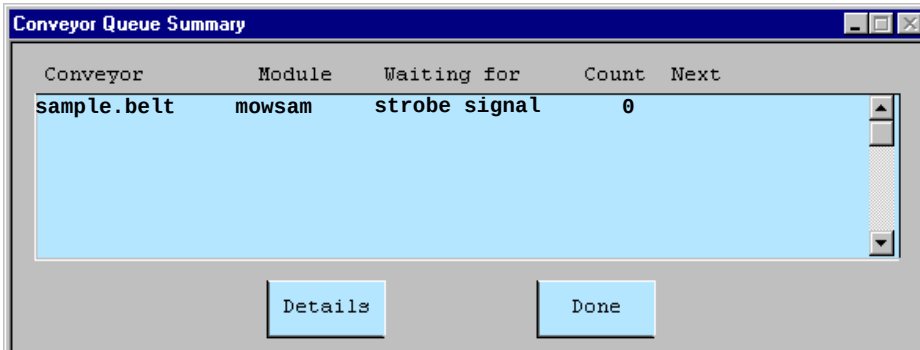


Figure 12-5

Conveyor Belt Status Display

The Conveyor column shows all active conveyor queues. The Waiting for column shows the next expected event (see Table 12-1).

In the above example, the queue is waiting for the strobe signal to indicate a new object on the belt. Count indicates how many objects in this queue. Next shows the name of the next object in the queue.

Choose **Details** to show the objects in the highlighted conveyor queue and their state.

Table 12-1

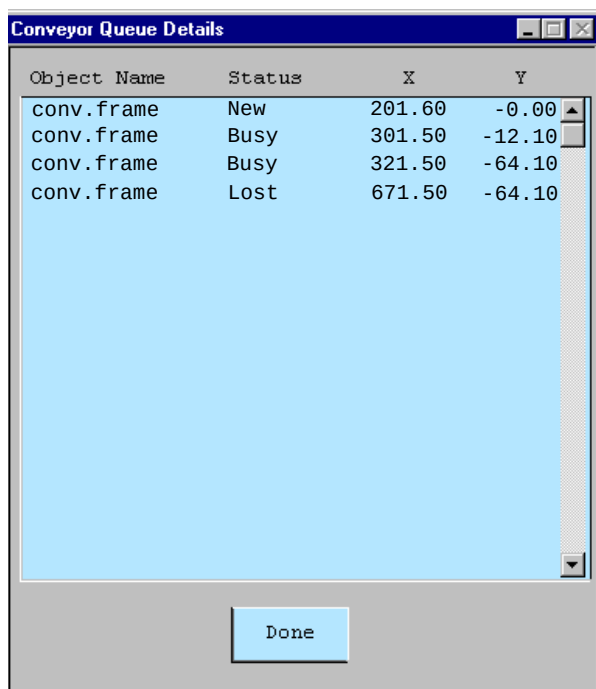
Waiting for Status Messages

Message	Meaning
advance delay	An indexing conveyor is waiting the delay time.
belt advance	An indexing conveyor is advancing.
belt distance	The queue is waiting for the sensor distance.
belt start	A tracking conveyor has been halted because a part passed the stop limit.
conveyor clear	The robot is below the clearance height and must move before the conveyor can index or start moving.
DISABLED	This conveyor frame is disabled, and objects will not be queued.
enable signal	The conveyor has not been enabled. The enable signal, which is defined in the conveyor record, is specified and deasserted. This message is displayed before the queue is accessed the first time.
external input	This message indicates that the queue is waiting for an external object from an upstream conveyor.

Table 12-1

Waiting for Status Messages (Continued)

Message	Meaning
picture clear	If a vision-equipped conveyor is taking pictures, this message indicates MotionWare is waiting for the robot to move away from the field of view.
queue empty	An indexing conveyor is waiting for its queue to empty so that it can be cycled.
queue lock	The queue is locked for processing.
queue scan lock	The queue is waiting for MotionWare to complete a scan of the queue before allowing additional entries to the queue.
sensor clear	An indexing conveyor queue is waiting for the sensor to be deasserted, indicating that the conveyor may advance.
sensor signal	The queue is waiting for the sensor signal to be asserted, indicating that a new object has been placed on the belt.
time interval	The queue is waiting for the sensor interval.
vision inspect	A picture has been taken and MotionWare is waiting for the results from a vision-inspection operation.
vision locate	A picture has been taken and MotionWare is waiting for the results from the vision-locate operation.

**Figure 12-6**
Conveyor Belt Object Status

In Figure 12-6, Object Name shows the names of the frame records that define the objects currently in the queue. An entry is displayed for each object that is detected.

Status shows the status of an object. (See Table 12-1.)

X and Y indicate the X/Y location of the object on the belt.

The status options are:

Status	Meaning
Busy	Robot is accessing object.
New	Object is queued, but has not been accessed.
Done	Object has been accessed, and its frame has been released.
Lost	Object has moved out of the belt window while being accessed.
Ignored	Object should be ignored by the current robot.

12.12 Conveyor Statements

The following statements are for use with conveyor belts.

GET_FRAME Statement

GET_FRAME is a special statement used with moving conveyor-belt frames.

The syntax for GET_FRAME is:

```
GET_FRAME --frame-- {1: --frame--} {2: --frame--}
                {3: --frame--} {4: --frame--} {5: --frame--}
                {6: --frame--} {7: --frame--} {8: --frame--}
                {WAIT --yes/no-- {WAIT_PATH --yes/no--}}
                {MATCH_CODE --variable--}{DISABLE --variable--}
                {OUTPUT --o_variable--}
```

This statement assigns objects from the queue manager to moving frames.

The first `--frame--` argument may be a variable frame, even if no object frames are specified. The remaining `--frame--` arguments should reference a conveyor-based reference frame.

The GET_FRAME MotionWare statement selects an object from a list of up to eight objects and assigns it to a frame. The statement finds the first object in the queue that is not disabled via the DISABLE bit-mask variable. A bit mask corresponding to the object assigned is returned in the OUTPUT variable.

Once an object is assigned to a frame, it must be released by using the RELEASE_FRAME statement or by moving to a location that does not have **Keep if success** or **Keep if error** selected on the location page.

If all of the object selection clauses (1: through 8:) are omitted, the GET_FRAME argument may be either an object definition frame or a variable frame. When the statement is

executed, an object frame is assigned the next matching object in the queue, or a variable frame is assigned the next object in the queue. If any of the object selection clauses are specified, the GET_FRAME argument must be a variable frame that receives information from the selected object in the queue.

The 1: --frame-- through 8: --frame-- arguments specify up to eight different objects that may be selected from the queue. The next object in the queue that matches any object specified by these arguments is returned in the variable frame in the GET_FRAME clause. The matching is further qualified by the MATCH_CODE and DISABLE arguments.

If WAIT is “yes”, this statement waits until a matching object is found in the queue. Otherwise, if no matching object is found, this statement exits immediately with an output value of zero.

If WAIT_PATH is “yes”, and the statement has to wait for an object, the robot enters the wait path specified in the conveyor record.

The MATCH_CODE clause is significant only if you are using the RELEASE_FRAME statement to set object code values.

- If the MATCH_CODE value is negative, no match processing occurs and all objects in the queue are processed as if the codes match.
- If the value is zero or positive, only objects in the queue with matching “code” values are processed. Others are ignored by this statement.
- If this clause is omitted, a match value of zero is assumed.

NOTE: By default, all objects in the queue have a code value of zero unless it has been changed by a RELEASE_FRAME statement.

The DISABLE variable contains a bit mask that indicates which frame object clauses (1: through 8:) should be ignored when searching for an object in the queue. The nth bit in the mask corresponds to the nth clause in the same manner as the OUTPUT --variable-- clause. More than one bit may be set in the mask. If this clause is omitted, a value of zero is assumed, which means that none of the object clauses are disabled.

The OUTPUT variable receives a bit mask indicating which object is selected. When no object frames are specified, this variable is set to -1 if an object is found, or 0 if no objects are found (and waiting is disabled).

Otherwise, the value is a bit mask that corresponds to the object clause that contains the object which is found, as follows:

Output	Clause
1	1:
2	2:
4	3:
8	4:
16	5:
32	6:
64	7:
128	8:

The DISABLE mask and OUTPUT variable can be used in combination to select a set of objects without regard to order, as shown in the example below:

```

SET disable = 0
RELEASE_FRAME vframe
WHILE disable <> 7
    GET_FRAME vframe = 1:object1 2:object2 3:object3 WAIT yes
    DISABLE disable OUTPUT status
    CASE status OF
        VALUE 1:
            MOVE TO location1
        VALUE 2:
            MOVE TO location2
        VALUE 4:
            MOVE TO location3
        ANY
            EXIT_LOOP
    END
    RELEASE_FRAME vframe
    SET disable = disable BOR status
END

```

The WAIT argument indicates whether the statement should wait if there are no objects queued. The WAIT_PATH argument indicates whether the robot should enter the wait path while waiting.

The MATCH_CODE argument is used with the export option to identify parts that have been exported from an upstream conveyor. The code is set by the RELEASE_FRAME statement (see below).

The remaining optional arguments are used with vision-based conveyors. See Chapter 15.

RELEASE_FRAME Statement

This statement releases moving frames assigned by the GET_FRAME statement.

The syntax for RELEASE_FRAME is:

```
RELEASE FRAME --frame-- {SET_CODE --variable--}
{IGNORE--yes/no--} {NEW--yes/no--}
```

The SET_CODE argument is set if the frame will be exported. It allows you to pass information about an object that has been exported to the downstream robot.

Select IGNORE if you want this object to be ignored by the current robot and to be optionally exported to a downstream robot. NEW indicates whether to return the object to the current queue as a new object.

CHECK_OBJECT Statement

The CHECK_OBJECT statement checks the status of an object that is assigned to a reference frame.

The syntax for CHECK_OBJECT is:

```
CHECK_OBJECT --frame-- {RESULT --o_variable--}
{CODE --o_variable--} STATUS --o_variable--
{NAME --string_variable--}
```

The --frame-- argument references the conveyor-based reference frame for which you want information.

The STATUS variable returns the following results:

Value	Description (for tracking conveyors)
0	No object assigned to this frame
1	Object is upstream
2	Object is the normal pickup window
3	Object is beyond the conveyor stop line
4	Object is beyond the conveyor pickup limit
5	Object is beyond the conveyor downstream limit
6	Object is lost

Value	Description (for indexing conveyors)
0	No object assigned to this frame
2	Object is in range
5	Object is out of range

If the STATUS variable is nonzero, the vision result is returned in the RESULT variable and the object code is returned in the CODE result. The optional NAME string variable receives the object name.

MATCH_OBJECTS Statement

The MATCH_OBJECTS statement checks that a minimum number of different objects are present in the conveyor queue and ready for acquisition.

This statement is normally used with an indexing conveyor, but it may also be used with a tracking conveyor that is equipped with a belt-stop signal. In this case, ☐ **Only stop busy objects** on the conveyor record must be left unchecked.

Up to four pairs of objects and object counts may be specified. A match occurs if the conveyor queue contains at least the specified number of each object, ready to be accessed.

The syntax for MATCH_OBJECTS is:

```
MATCH_OBJECTS 1:--variable-- of --frame-- {2:--variable-- of
--frame-- {3:--variable-- of --frame-- {4:--variable-- of
--frame--}}}{MATCH_CODE --variable--}{WAIT --yes/no--}
{INDEX --yes/no-- {TIMEOUT --variable--} {IGNORE --yes/no--}}
{OUTPUT --o_variable--}{WAITING --o_variable--}
```

If the optional MATCH_CODE clause is specified, all objects must have a corresponding matching code in order to be counted.

If WAIT is “no” and INDEX is “no”, this statement samples the object queue once and returns match information in the OUTPUT and WAITING variables.

If WAIT is “yes” and INDEX is “no”, this statement waits until either all objects match or until the conveyor manager has completed processing all vision information. The OUTPUT and WAITING variables show the results.

If WAIT is “no” and INDEX is “yes”, this statement samples the object queue once and returns match information in the OUTPUT and WAITING variables. In addition, if the conveyor is in wait state, the conveyor indexes or advances once. The OUTPUT and WAITING variables show the results.

If WAIT is “yes” and INDEX is “yes”, this statement samples the object queue and repeatedly indexes or advances the conveyor until all the objects match or until the TIMEOUT value is reached. TIMEOUT is specified in seconds. If TIMEOUT is omitted, this statement may cycle the conveyor indefinitely. The OUTPUT and WAITING variables show the results.

OUTPUT is an output variable that receives a bit mask indicating which items, if any, matched the count specified. A zero value indicates that no matches occurred.

WAITING is an output variable that is set to TRUE if the conveyor is waiting for objects to be removed from the queue. Otherwise, it is set to FALSE.

The optional clauses WAIT, INDEX, and IGNORE are assumed to be “no” if they are omitted.

RESUME_CONVEYOR Statement

The RESUME_CONVEYOR statement attempts to resume a conveyor that is in a wait state. If the conveyor is *not* in a wait state, this statement does nothing.

- An indexing conveyor is in a wait state when it is waiting for its object queue to be emptied. This statement empties the queue by releasing all objects that are not marked “busy”.
- A tracking conveyor is in a wait state when it is stopped due to “new” objects beyond the belt-stop line. This statement changes the state of all such objects to “done” or “ignored” depending on the IGNORE clause. It also sets the object code for any changed objects to the value specified by the SET_CODE clause.

The syntax for RESUME_CONVEYOR is:

```
RESUME_CONVEYOR --conveyor-- {WAIT --yes/no--}
                  {SET_CODE --variable--} {IGNORE --yes/no--}
```

If the WAIT clause is “yes”, this statement first waits for the conveyor to become idle before proceeding. If the conveyor never becomes idle, this statement may wait indefinitely.

If the SET_CODE clause is specified, the statement sets the object code for any changed objects to the specified value.

If the IGNORE clause is “yes”, all “new” objects beyond the belt-stop line are set to “ignored”. See the paragraph on tracking conveyors, above.

12.13 Conveyor Tracking Example

This section provides a generic example of conveyor tracking without vision. After completing this example, you should have a basic understanding of how conveyor tracking works.

NOTE: It is not necessary to complete the steps in any particular order, as long as they are all completed.

This example assumes a conveyor and robot configuration similar to the one in the illustration below:

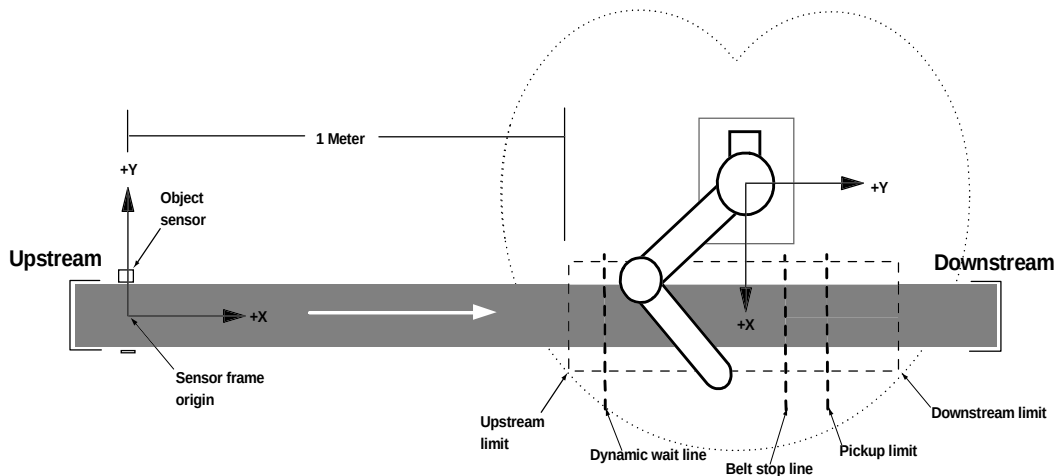


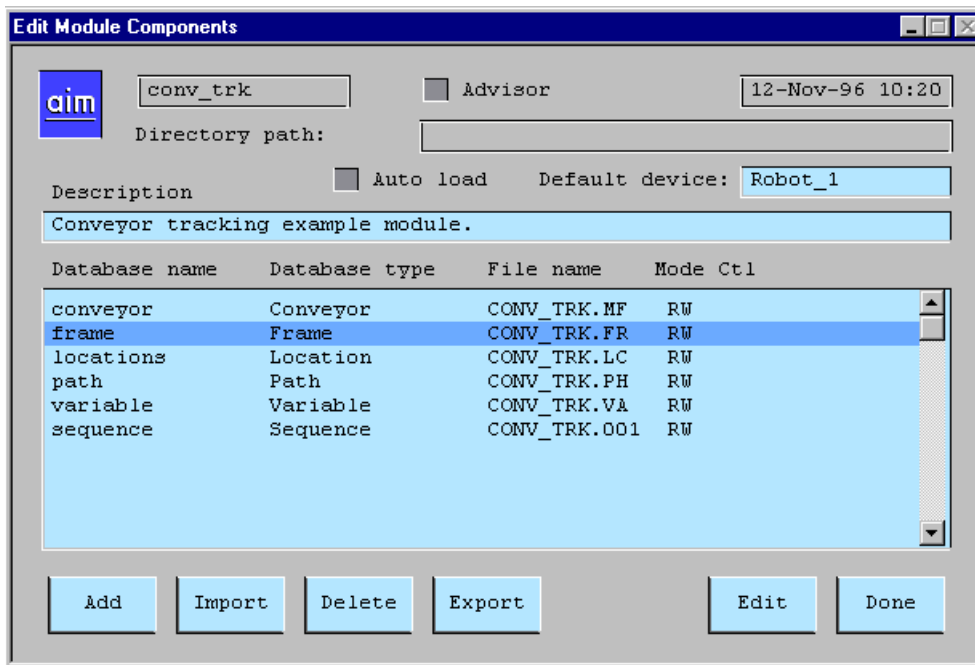
Figure 12-7
Example Conveyor Tracking Setup

Create a Resource Module

Create a Resource Module called “conv_trk” (refer to Chapter 3 for details). This module should include the following databases:

- locations
- sequence
- conveyor
- path
- variable
- frame

The finished resource module should look similar to the following:



Create a Conveyor Record

Create a conveyor record called “conv_rec” (refer to Figure 12-3 for details).

1. Enter “conv_rec” for the record name. You also can enter a description for the record.
2. Select the device associated with this conveyor, item ❶.
3. Select ☒ **Yes** for the Continuous option, item ❸.
4. Specify ☒ **Sensor** as the Object source.
5. Choose **Define**, item ❹, to enter any digital I/O signals used by this conveyor. Enter the sensor signal name or number in the Sensor option of the Control Signals, item ❹.
6. Select any export options, item ❺.
7. Define the continuous conveyor parameters (refer to the illustration on page 141). All parameters with a **Here** button can be defined by placing the robot in the appropriate location and then choosing the corresponding **Here** button.

Create a Frame Record

Create a Frame record called “conv_frame” (refer to “Creating Reference Frames” on page 100). Make sure this record references the Conveyor record (created above) and uses the Object Definition to make this frame relative to a moving conveyor belt.

1. Enter “conv_frame” for the record name. You can also enter a description for this record.
2. Select ☒ **Object Definition** as the frame type.
3. Select ☒ **Non Vision** as the object definition.
4. Double-click the Conveyor data box and retrieve the conveyor record “conv_rec” (created previously) from the pick list.
5. If you want access to this object to be based on the Object Enable signal, select ☒ **Use object enable signal**.

Create a Location Record

Create Location record called “conv_loc” (refer to “Editing Locations” on page 74).

The screenshot shows the 'locations (conv_trk)' window with the following details:

- Title Bar:** locations (conv_trk)
- Menu Bar:** Go, Seek, Edit, Help
- Top Section:**
 - Record Name: conv_loc (1 of 1)
 - Device: Robot_1
 - Date/Time: 12-Nov-96 12:00
 - Buttons: Optimize, Approach/Move, Depart
- Coordinate Fields:** X, Y, Z, y, p, r (all blank)
- Table Headers:** Line/Joint, Speed, Rot. Speed, in/sec, mm/sec, Coarse/Fine, Accel, Decel, Prof
- Table Rows:**
 - Location:** Line/Joint: 100, Speed: 0, Rot. Speed: [blank], in/sec: [blank], mm/sec: [blank], Coarse/Fine: [blank]/[checked], Accel: 100, Decel: 100, Prof: 0
 - Approach:** Line/Joint: 100, Speed: 0, Rot. Speed: [blank], in/sec: [blank], mm/sec: [blank], Coarse/Fine: [blank]/[blank], Accel: 100, Decel: 100, Prof: 0
 - Depart:** Line/Joint: 100, Speed: 0, Rot. Speed: [blank], in/sec: [blank], mm/sec: [blank], Coarse/Fine: [blank]/[blank], Accel: 100, Decel: 100, Prof: 0
- Buttons:** Here, Strategies, Pallet, Teach, Configuration
- Options:**
 - ☐ Stop after motion
 - Retry Count: 0
 - Settling Time: 0.00
 - Dwell Time: 0.00
- Reference Frame:**
 - ☒ World, ☐ Named
 - Named Frame: conv_frame
 - ☐ Keep if success, ☐ Keep if error
- Input Signals:** Three empty input boxes.
- Output Signals:** Six empty output boxes.

1. Enter “conv_loc” for the record name.
2. Make this record relative to the Frame record “conv_frame” created previously. (Select ☒ **Named** [in the Reference Frame box] to display the data box requiring a frame record from the frame database. Double-click on the field and select “conv_frame” from the pick list.)
3. Leave the location value (X, Y, Z, y, p, r) blank. Since tracking conveyor reference frames move, this value can be taught only during walk-thru training.

Calibrate the Tracking Conveyor

The tracking conveyor must be calibrated to establish the relationship between encoder¹ ticks and belt travel. Calibration also establishes the relationship between the belt direction and the world coordinate system. Refer to the tracking conveyor calibration procedure described on page 152.

¹ To perform conveyor tracking without a camera or indexed conveyor, you must use an encoder. Adept VJI modules support incremental encoders without modifications (see the *Adept MV Controller User's Guide*, pages 114 and 115).

Install an Object Sensor

Install an object sensor (through beam type) on the conveyor and wire the input signal either into the SIO or DIO board (pin outs are described in the *Adept MV Controller User's Guide*, pages 66 – 68 for SIO and page 129 for DIO). Make sure that you test the functionality of the sensor using the **I/O ➔ Digital Signals** menu page.

Create a sensor frame for the location of the object sensor (this is done in the Conveyor record by choosing **Sensor Frame**). Remember that the object sensor does not have to be located within the robot's work envelope. If the sensor is positioned like the one in Figure 12-7, add -1000 mm to the Y value and then recompute the Frame Value.

Create a Sequence

Create a sequence like the one below. It is better to keep the sequence simple and straightforward at first. Once you've proven that you can successfully track an object and pick it up, you can always add more functionality to the sequence.

```
1.  begin:
2.    MOVE TO safe
3.    MOVE ALONG pick_path TO pick_conveyor
4.    REPEAT 24
5.      MOVE ALONG place_path TO place_pal
6.      MOVE TO pick_conveyor
7.      IF pallet_full
8.        GOTO pal_full_reset
9.      END
10.   END
11.  pal_full_reset:
12.    IO OUTPUT pallet_reset
13.    GOTO begin
```

While still in the sequence editor, double-click on the "safe" location to create a SAFE location record. Move the robot to a clear area in the workcell and choose **Here**. Return to the sequence editor and repeat this procedure for the "place_pal" location. After you create the PLACE_PAL location record, return to the sequence editor.

Now you must create a record for the "pick_conveyor" location. However, you cannot teach this location using the **Here** button. Instead, this location must be taught by executing the sequence in walk-thru training mode. While the sequence is executing, you will receive an error message that asks you to teach the "pick_conveyor" location. See "Single-Access Operation" on page 151.

MotionWare Statements

13

13.1 Introduction	168
13.2 General MotionWare Statements	169
MOVE Statement	169
How MOVE Uses the Databases	170
Things to Remember	172
MOVE_REL Statement	172
DRIVE Statement	173
SET_FRAME Statement	173
Things to Remember	174
SET_TOOL Statement	174
STOP_ROBOT Statement	174
WAIT_UNTL_ROBOT Statement	174
RESET_PALLET	176
MESSAGE	176
SET_RESPONSE	176
13.3 General AIM Statements	177
Using Expressions in Statements	177
Conditional and Looping Structures	178
CALL Statement	178
IF Statement	178
END Statement	179
ELSE Statement	179
FOR Statement	180
REPEAT Statement	181
WHILE Statement	181
EXIT_LOOP Statement	182
NEXT_LOOP Statement	182
CASE Structure (Includes VALUE and ANY Statements)	182
GOTO Statement	183
SET Statement	183
SETS Statement	183
RETURN Statement	185
13.4 Sequence Control Statements	185
LOAD Statement	185
PAUSE_TASK Statement	185
RESUME_TASK Statement	186
SELECT_TASK Statement	186
START_TASK Statement	186
STOP_TASK Statement	186
UNLOAD Statement	187
WAIT Statement	187
WAIT_VAR Statement	187

13.1 Introduction

There are several classes of MotionWare statements:

Conveyor Statements (detailed in Chapter 12):

CHECK_OBJECT	GET_FRAME
MATCH_OBJECTS	RELEASE_FRAME
RESUME_CONVEYOR	

Digital I/O Statements (detailed in Chapter 9):

IO	IO_LIST
IO_LIST_VAR	WAIT_FOR
WAIT_UNTIL	

General AIM Statements (detailed in this chapter):

CALL	CASE..VALUE..ANY
ELSE	END
EXIT_LOOP	FOR
GOTO	IF
NEXT_LOOP	REPEAT
RETURN	SELECT_VAR_REC ¹
SET	SETS
WHILE	

General MotionWare Statements (detailed in this chapter):

DRIVE	MESSAGE
MOVE	MOVE_REL
RESET_PALLET	SET_FRAME
SET_RESPONSE	SET_TOOL
STOP_ROBOT	WAIT_UNTIL_ROBOT

MotionWare Statements Requiring Vision (detailed in Chapter 14):

INSPECT	PICTURE
REFINE_FRAME	REFINE_TOOL

Sequence Control Statements (detailed in this chapter):

LOAD	PAUSE_TASK
RESUME_TASK	SELECT_TASK
START_TASK	STOP_TASK
UNLOAD	WAIT
WAIT_VAR	

¹ This statement is detailed in Chapter 6.

13.2 General MotionWare Statements

See Chapter 4 for a description of how sequences are created and edited. See section 4.4 on page 60 for details on the source of statement arguments.

MOVE Statement

The MOVE statement executes the basic robot tasks of MotionWare.

The syntax for MOVE is:

```
MOVE {{APPROACH --path--} FROM --location--} {ALONG --path--}
TO --location-- {DEPART --path--} {USING --tool--}
{{ALONG --path--} REJECT --location-- {RETURN--path--}}
{WAIT_AT --location--} {OK_SIGNAL --o_variable--}
```

The sequence of operations performed by this statement is:

1. If a tool transformation is specified (USING--tool--), apply that tool to describe the current robot gripper. Otherwise, leave the tool transformation unchanged.
2. If an approach path is specified (APPROACH--path--), move along that path to the FROM location.
3. If a FROM location is specified (FROM--location--), move to that location and execute the location strategy routine or sequence specified in the location record.

If specified in the location record, the approach and depart strategies will be executed at the beginning of the approach and beginning of the depart move for that location.

4. If the first along path is specified (ALONG--path--), move along that path to the TO location.
5. Move to the TO location (TO--location--) and execute the location strategy routine or sequence specified in the location record.

If specified in the location record, the approach and depart strategies will be executed at the beginning of the approach and beginning of the depart move for that location.

6. If a depart path is specified (DEPART--path--), then depart the TO location along that path.

The remaining steps are executed only if an error occurs and the operator chooses the Retry Statement, Skip Statement, or Skip Cycle button from the debug control panel or the strategy routine returns **rn.opr.fail**.

7. If a second along path (ALONG--path--) is specified, move along that path to the reject location.
8. Move to the reject location (REJECT--location--) and execute the strategy or sequence specified in the location record.
9. If the return path is specified (RETURN--path--), then depart from the reject location along that path.

If the TO or FROM location is relative to a moving conveyor, the WAIT_AT location specifies part of the location for the robot to wait until the part is within the belt window.

The complete wait location is determined as follows:

- If there are no “new” objects in the queue, use the path specified in the conveyor record.
- If the object is upstream of the belt window, get the location Z, APPRO, ACCEL, configuration options, speed, and motion parameter values from the WAIT_AT argument. Use the **Dynamic wait line** from the conveyor record for the position in the belt direction of travel. Use the location of the object in the queue to determine the robot position lateral to belt (perpendicular to direction of belt travel) and the object rotation for the robot orientation.

This strategy allows you to place the robot in the most advantageous position to access the next object.

The OK_SIGNAL is output signal set to TRUE if the statement completes all defined arguments through the DEPART argument. Otherwise, it is set to FALSE.

How MOVE Uses the Databases

Table 13-1 shows the statement clause, the databases accessed by that clause, and the information retrieved from each database.

Table 13-1
MOVE Statement Databases

Statement	Database Accessed	Information Retrieved
APPROACH path	Path	Path to use moving from current robot location to the FROM location Entry point nearest the current robot location Exit point nearest the location being moved to Transit locations between the entry and exit points
FROM location	Location	The robot location, including: Approach and depart heights Arm configuration Reference frame Motion parameters for all moves Gripper I/O Whether the location is a pallet location, and the parameters for the pallet The location/approach/depart strategies
	Frame	If used, a named reference frame
ALONG path	Path	Path to use moving from the FROM location to the TO location Entry point nearest the current robot location Exit point nearest the location being moved to Transit points between the entry and exit points

Table 13-1
MOVE Statement Databases (Continued)

Statement	Database Accessed	Information Retrieved
TO location	Location	The robot location, including: Approach and depart heights Arm configuration Reference frame Motion parameters for all moves Gripper I/O Whether the location is a pallet location, and the parameters for the pallet The location/approach/depart strategies
	Frame	If used, a named reference frame (can be a conveyor based reference frame)
DEPART path	Path	Path to use when leaving the TO location Entry point nearest the current robot location Exit point nearest the location being moved to Transit points between the entry and exit points
USING tool	Tool	A tool transformation to use for all robot motions and to apply to locations taught during walk-thru training
ALONG path	Path	Path to use moving to the reject location (when the operator has indicated the statement should be reexecuted) Entry point nearest the current robot location Exit point nearest the location being moved to Transit points between the entry and exit points
REJECT location	Location	The part reject location, including: Approach and depart heights Arm configuration Reference frame Motion parameters for all moves Gripper I/O Whether the location is a pallet location, and the parameters for the pallet The location/approach/depart strategies
	Frame	If used, a named reference frame
RETURN path	Path	Path to use when leaving the REJECT location Entry point nearest the current robot location First exit point after the entry point Transit locations between the entry and exit points
	Frame	If used, a named reference frame

Table 13-1
MOVE Statement Databases (Continued)

Statement	Database Accessed	Information Retrieved
WAIT_AT location	Location	The robot location, including: Arm configuration Reference frame Motion parameters for all moves Location Z height
	Frame	If used, a named reference frame
OK_SIGNAL variable	Variable	Digital signal to set when the statement completes successfully.

Things to Remember

All locations must be taught with the proper tool transformation in effect.

The robot gripper is not automatically closed at the FROM location and opened at the TO location. All robot gripper activity is controlled by the I/O signals specified in the location record (as long as **rn.mw.standard** is the acquire/release strategy).

A MOVE statement does not necessarily acquire a single object and then place that object. A single MOVE statement can acquire two parts, or multiple MOVE statements can be used to load end effectors, which can acquire multiple parts. The digital I/O used by the multiple grippers must agree with I/O signals specified in each location record. Similarly, multiple MOVE statements can be used to unload the multiple grippers.

MOVE_REL Statement

This statement allows you to move a robot relative to a defined location. You can specify a value for the distance in the tool X, Y, or Z planes (DX, DY, or DZ) or a value for rotation (RX, RY, or RZ).

The syntax is:

```
MOVE_REL WRT --segment-- {DX --variable--}  
{DY --variable--} {DZ --variable--} {RX -- variable--}  
{RY --variable--} {RZ --variable--} {SPEED --variable--}  
{ACCEL --variable--} {DECEL --variable--} {NULLING --mode--}
```

In **--segment--** specify whether this move will be relative to the approach, depart, or final location. In DX, DY, DZ, RX, RY, and RZ, specify the magnitude of the relative move. In the SPEED, ACCEL, and DECEL arguments, specify motion parameters for this move. The move will be relative to the current or most recently completed move. The **--segment--** determines whether the move is relative to the approach, location, or depart segment of a move.

The NULLING parameter can have the following values:

- "coarse" Null using V+ COARSE tolerance
- "fine" Null using V+ FINE tolerance
- "none" Null using V+ NONULL mode

If NULLING is not used, the nulling mode set by the original motion segment remains in effect.

This statement is intended primarily for strategy routines.

DRIVE Statement

This statement allows you to move a single joint of the robot. You can specify a value for the joint, distance, and speed (with optional parameters for acceleration, deceleration, and nulling).

The syntax is:

```
DRIVE JOINT --variable-- DISTANCE --variable--  
      SPEED --variable-- {ACCEL --variable--}  
      {DECEL --variable--} {NULLING --mode--}
```

In JOINT, specify the robot joint to move. In the SPEED, ACCEL, and DECEL arguments, specify motion parameters for this move. The NULLING parameter can have the following values:

```
"coarse"  Null using V+ COARSE tolerance  
"fine"    Null using V+ FINE tolerance  
"none"    Null using V+ NONULL mode
```

SET_FRAME Statement

```
SET_FRAME --frame-- EQUAL_TO {FRAME --frame--}  
{INVERSE --frame--} {FRAME --frame--} {VISION_FRAME --vision--}  
{SAVE_PREVIOUS --yes/no--} {OK_SIGNAL --o_variable--}
```

The VISION_FRAME --vision-- argument can be used only if your AIM system includes the optional AIM Vision Module.

The operation performed by this statement is as follows:

1. If specified, the optional frame argument values are read from the Frame database. ({FRAME --frame--} {FRAME --frame--})
2. If specified, {INVERSE --frame--} is read and inverted.
3. If specified, the optional vision operation is evaluated. The resulting vision frame value is converted to robot coordinates. ({VISION_FRAME --vision--}) Depending on the vision picture specification, this vision operation may cause the robot to move in order to take a picture.

If yes is specified, the previous image data will be used to generate the vision portion of the frame, no new pictures will be taken. ({SAVE_PREVIOUS --yes/no--})
4. The specified frame values are combined and stored in the record specified by the required frame argument. If no values were specified, this record is set to NULL. If an error occurred in computing any of the argument values, this record is set to undefined. If two or more values were combined, the result is normalized before it is stored. (SET_FRAME --frame--)
5. If the output signal is specified and no errors occurred in computing the specified values, the status output signal is set to TRUE. Otherwise, it is set to FALSE. ({OK_SIGNAL --o_variable--})

Things to Remember

- The required `--frame--` argument should reference records in which Fixed is selected.
- The optional `--frame--` arguments may reference records in which Object Definition is selected.
- The vision operation should return a vision frame result, which will be converted to robot coordinates.
- The optional path record specified in the picture record contains the parameters for moving to the picture-taking location. Parameters include speed, motion type, and similar attributes.
- See Chapter 8 for details on reference frames.

SET_TOOL Statement

```
SET_TOOL {--tool--}
```

The SET_TOOL statement sets a new robot tool transformation. This tool will be in effect either until another SET_TOOL statement is executed or a MOVE statement with a USING clause is executed. There is one optional argument to the statement, a record from the tool database. A SET_TOOL statement without an argument sets the tool transformation to NULL.

If the current tool is different from that specified, continuous-path robot motion is temporarily disabled as the new tool value is set. That is, a break in continuous path occurs.

STOP_ROBOT Statement

```
STOP_ROBOT {AFTER_MOTION --yes/no--}
```

If the `--no--` is selected or the argument is omitted, then the robot is stopped immediately. If `--yes--` is selected, then the robot stops after the current motion completes. This mode is useful only for canceling belt tracking.

WAIT_UNTL_ROBOT Statement

This statement allows you to pause a sequence until one of the following occurs. (These options can be limited by a time-out value.)

- The robot stops and settles
- The robot moves within a specified distance of a location
- The robot moves outside a specified distance from a location
- The robot completes a percent of its current move
- The robot has less than a specified amount of time left to complete a move

This statement is intended primarily for strategy routines or the IO_Control task.

The syntax is:

```
WAIT_UNTL_ROBOT {DEVICE --device--} {STOPPED --yes/no--
{SETTLE --constant--}} {< --constant-- {FROM --location--}
{ALONG --component--}} {> --constant-- FROM --location--
{ALONG --component--}} {%_COMPLETE --constant--}
{TIME_LEFT --constant--} {TIMEOUT --constant--
{TIMEOUT_STATUS --o_variable--}
{MOTION_STATUS --o_variable--}}
```

MOTION_STATUS variable is set to indicate the final motion status as follows:

- 0 If the current motion is running or completed normally.
- 1 If the current or previous motion completed abnormally.
- 2 If the robot has transitioned to another segment.
The settling time may have been skipped.
The TIMEOUT or %_COMPLETE condition may have terminated early.

For example, if E_STOP occurs during a motion while WAIT_UNTL_ROBOT is executing, the MOTION_STATUS will be:

- 1 If the operator chooses to skip the current motion and leaves the robot stopped.
- 2 If the operator retries the current motion.

The operations performed by this statement are:

1. If a --device-- is specified, select that robot for use by this routine. If no device is specified, use the current robot. (Note that a different robot cannot be selected if this task is controlling a robot.)
2. If yes is selected for the STOPPED clause, wait until the robot is stopped. After the robot stops, wait the optional settling time (specified in SETTLE --constant--). The time is in 1/100th of a second. Ignore all remaining clauses except "TIMEOUT --constant--".
3. If the < --constant-- clause is specified, and the FROM --location-- clause is specified, wait until the robot is less than ("<") the specified distance, in millimeters, from that location. If the FROM --location-- clause is not specified, use the distance to the end of the current motion segment. If a Cartesian axis is specified in the optional ALONG --component--, the difference of the X, Y, or Z components of the location is used for comparisons. Otherwise, the shortest straight-line distance is used. Shortest distance is always a positive value, but the distance along a Cartesian axis may be positive or negative. HPS locations may not be used for the FROM --location--.
4. If the > --constant-- clause is specified, wait until the distance from the specified location is greater than (">") the specified value, in millimeters. If a Cartesian axis is specified in the optional ALONG --component--, the difference of the X, Y, or Z components of the location is used for comparisons. Otherwise, the shortest straight-line distance is used. Shortest distance is always a positive value, but the distance along a Cartesian axis may be positive or negative. HPS locations may not be used for the FROM --location--.
5. If the %_COMPLETE --constant-- clause is specified, wait until the current robot motion is complete by at least the specified value (from 1 to 100).
6. If the TIME_LEFT --constant-- clause is specified, wait until the time left in the current robot motion is less than the specified value, in hundredths of a second.

7. If the optional `TIMEOUT --constant--` clause is specified, check for a time-out condition when waiting for any of the above. The time-out value is specified in hundredths of a second.
8. If a time-out occurs, set the optional `TIMEOUT_STATUS --o_variable--` variable to TRUE and exit without error.

RESET_PALLET

This statement allows you to reset the pallet digital signals. The `--location--` argument specifies which location record should have its pallet signals reset.

The syntax is:

```
RESET_PALLET --location--    {SET_SIGNALS --yes/no--}  
                             {CLEAR_SIGNALS --yes/no--}
```

If the `SET_SIGNALS` argument is “yes”, all the associated pallet signals are set. If the `CLEAR_SIGNALS` clause is “yes”, all the associated pallet signals are cleared. If both are specified, set overrides clear.

MESSAGE

This statement displays a message on the operator control panel.

```
MESSAGE --variable-- {TEXT --string--} {0: --constant--}  
                  {1: --constant--} {2: --constant--} {3: --constant--}  
                  {4: --constant--} {5: --constant--} {6: --constant--}  
                  {RESPONSE --o_variable--}
```

The `MESSAGE --variable--` is a standard AIM error number (or variable record that specifies an error number), which is in the error database. The text specified in `TEXT --string--` will be appended to this message and displayed on the task control panel.

The `--constant--` arguments correspond to the seven standard operator response buttons on the Task Control Panel. A value of 0 indicates that the corresponding button is not valid a value of -1 indicates that the standard AIM labels should be applied to the button and a value greater than 0 indicates the button label should be taken from the corresponding standard AIM message number in the error database. If any nonzero values are specified for these seven arguments, the sequence pauses until an operator response button is pressed and the number of the button pressed is returned in the `RESPONSE` variable.

If all arguments are blank or 0, AIM handles errors in the normal fashion and does not wait for an operator response.

SET_RESPONSE

This statement allows an executing sequence to behave as if one of the operator response buttons had been pressed. The `--variable--` should have a value of 0 to 6, corresponding to the 7 standard operator response buttons. If an invalid response code is received, an error is issued.

The syntax is:

```
SET_RESPONSE --variable--
```

13.3 General AIM Statements

The statements in this section allow you to conditionally execute blocks of statements, loop repeatedly through blocks of statements, and start execution of other sequences.

Using Expressions in Statements

Several AIM statements allow expressions to be entered in their argument lists. For example: the SET, SETS, IF, and WHILE statements all accept expressions.

AIM expressions are combinations of operators, variables, and constants. No parentheses are allowed, so the precedence of the operators determines the value of the expressions.

The unary operators ("-", "COM", "NOT") always have the highest precedence. Therefore, expressions like:

NOT a AND b

are evaluated as (NOT a) AND b.

Each binary operator has its own precedence. Higher precedence means that the operations occur first. If operators have the same precedence, they are evaluated from left to right.

Operators	Precedence
* /	10
+ -	9
< > =< >=	8
== <>	7
BAND BAND_COM	6
BXOR	5
BOR BOR_COM	4
AND AND_NOT	3
XOR	2
OR OR_NOT	1

For example, the expression:

`a > b AND c > d OR a+b < c`

is evaluated as:

`((a > b) AND (c > d)) OR ((a+b) < c)`

Conditional and Looping Structures

Conditional and looping structures allow you to specify under what conditions and how many times blocks of statements will execute. Conditional and looping structures may be very simple and straightforward or they can use complex arithmetic expressions and relational logic. The examples in the following sections present the most common uses of the various structures. A complete understanding of the complex ways that the structures can be used requires that you understand relational logic. (The *V+ Language User's Guide* describes the basics of relational logic.)

All the structures share the following common attributes:

The block of statements that will be executed begins immediately after the structure statement and ends with an END statement. There must be an END statement for each control structure statement; multiple control structures cannot be ended with a single END statement.

Structures may be nested within each other.

The variables used in a looping statement (FOR, REPEAT) should *never* be changed by any statements within the loop. Also, during execution of the loop, any other programs that might have access to global variables used in the looping statement should not change those variables.

In all computer systems, numbers are internally represented in binary format. This internal format may involve some rounding of the actual number since the internal representation may not exactly represent base 10 numbers. This internal representation can lead to unexpected behavior when comparing real values (such as 3.1467) since the displayed value of two numbers may be identical but their internal representation may be different. When using variables in statements that test for equality, use integer values to avoid this potential problem.

CALL Statement

The CALL statement allows you to call one sequence from within another sequence. When a CALL statement is encountered, the currently executing sequence (calling sequence) will suspend operation and the new sequence (called sequence) will begin. When the called sequence completes, the calling sequence will resume execution at the next statement line.

Sequence calls can be nested, and a called sequence itself can call another sequence. The **Stack** button on the Task Control Panel will show you how many calls have been made and the point in the sequence that the calls were made. Only sequences in the assigned module can be executed using the CALL statement.

IF Statement

The IF statement allows you to conditionally execute a block of statements.

The syntax is:

```
IF {--uopr--} --variable-- {--opr-- --variable-- {--opr--
--variable-- {--opr-- --variable-- {--opr-- --variable--
{--opr-- --variable--}}}}}
```

To execute a block of statements if a variable has a certain value, specify the variable in the first `--variable--` clause, select “==” as the first `--opr--`, and specify the required value as the second `--variable--`:

```
IF x == 3
```

If you want to execute a block of statements based on the value of two variables, you could use one of the following statements:

```
IF x == 3 AND y == 7
IF x == 3 OR y == 7
```

The first example executes the statement block if x has a value of 3 and y has a value of 7—both variables must have the indicated values. In the second example, if either variable has the indicated value, the statement block will execute.

To create a conditional block that depends on the state of a digital signal, select a variable that has Input signal or Output signal selected:

```
IF in_sig
IF NOT in_sig
```

In the first case, the statement block will execute if the signal is on. In the second case, it will execute if the signal is off.

END Statement

A statement block is terminated with an END statement. In the following sequence, the first statement will execute, statements 3 and 4 will execute only if x has a value of 7, and statement 6 will execute normally.

```
1. STATEMENT one
2. IF x == 7
3.     STATEMENT three
4.     STATEMENT four
5. END
6. STATEMENT six
```

ELSE Statement

An ELSE statement is always used in conjunction with an IF statement. The ELSE statement specifies a block of statements to execute when the statements specified by the preceding IF statement are not executed. The ELSE statement can also specify conditions to control execution of statements within the ELSE block.

The syntax of the ELSE statement is:

```
ELSE {IF {--uopr--} --variable-- {--opr--
--variable-- {--opr-- --variable-- {--opr-- --variable--
{--opr-- --variable-- {--opr-- --variable--}}}}}
```

In the following example, statement 2 will execute if x has a value of 7, and statement 4 will execute if it has any other value:

```
1. IF x == 7
2.     STATEMENT two
3. ELSE
4.     STATEMENT four
5. END
```

In the following example, statement 2 will execute if x == 7, and statement 4 will execute if x has any other value and y == 3. If both x and y do not have the specified values, neither statement will execute:

```
1. IF x == 7
2.     STATEMENT two
3. ELSE IF y == 3
4.     STATEMENT four
5. END
```

FOR Statement

The FOR statement allows you to execute a block of statements a specified number of times.

```
FOR --o_variable-- = {--uopr--} --variable-- {--opr--
--variable--} TO {--uopr--} --variable-- {--opr-- --variable--}
{STEP --constant--}
```

FOR statements can become complex. Following are several examples beginning with simple FOR loops and proceeding to complicated FOR loops.

The following example executes a block of code four times:

```
FOR loop_var = 1 TO 4
```

In the above example, **loop_var** (which must be an AIM variable, V⁺ variable or **ai.ctll[]** value in the Variable database) will increment once each time the loop is executed. If any other statements within the loop access the variable **loop_var**, the variable will have the value of the current cycle of the loop.

You could make the FOR loop “count by twos” by including the STEP argument:

```
FOR loop_var = 2 TO 4 STEP 2
```

The above example would execute twice with **loop_var** having the value of 2 and 4 during successive loops.

You can count from a negative number by using the --uopr-- argument:

```
FOR loop_var = -2 TO 4 STEP 2
```

Double clicking --uopr-- will give you the option of “-”, NOT, or COM. The “-” is the only selection that would normally be used in a FOR statement. You also can make the FOR statement count backwards by giving the STEP argument a negative value (note the relationship of the “=” and TO argument values):

```
FOR loop_var = 4 TO -2 STEP -1
```

You can make the “=” and TO argument values rely on a calculation by using the --opr-- and --variable-- arguments:

```
FOR loop_var = 4 * x TO -2 STEP -1
```

REPEAT Statement

The REPEAT statement allows you to repeat a block of statements a given number of times. The arguments to REPEAT allow you to evaluate a complex expression when deciding how many times to repeat a block of statements.

The syntax is:

```
REPEAT {--uopr--} --variable-- {--opr-- --variable--
      {--opr-- --variable-- {--opr-- --variable-- {--opr--
      --variable-- {--opr-- --variable--}}}}}
```

To simply repeat a block of statements a given number of times, specify that number for the first --variable--:

```
REPEAT 4
```

To repeat a loop 4 times plus the value of the variable x, select “+” as --opr-- and the record x as the second --variable--:

```
REPEAT 4 + x
```

The block of statements included in a REPEAT loop is identified by an END statement. In the following example, statements 2 through 6 will be executed five times:

```
1. REPEAT 5
2.     STATEMENT two
3.     STATEMENT three
4.     STATEMENT four
5. END
```

WHILE Statement

The WHILE control structure causes a block of statements to be indefinitely executed as long as the specified conditions are correct.

The syntax for the WHILE statement is:

```
WHILE {--uopr--} --variable-- {--opr-- --variable--
      {--opr-- --variable-- {--opr-- --variable-- {--opr--
      --variable-- {--opr-- --variable--}}}}}
```

In the following example, statements 2 and 3 will execute only if signal **in_sig1** is on or signal **in_sig2** is off when statement 1 is encountered, and they will continue to execute as long as **in_sig1** is on or **in_sig2** is off:

```
1. WHILE in_sig1 OR_NOT in_sig2
2.     STATEMENT two
3.     STATEMENT three
4. END
```

EXIT_LOOP Statement

An EXIT_LOOP statement will cause a REPEAT, FOR, or WHILE loop to terminate and sequence execution to continue at the next statement following the END statement that closes the REPEAT or WHILE loop.

The syntax for an EXIT_LOOP statement is:

```
EXIT_LOOP {--constant--}
```

This optional argument specifies the number of loops to exit. If an EXIT_LOOP is within a REPEAT loop that is nested within a WHILE loop, specifying 2 for the --constant-- argument will result in terminating the execution of both loops.

An EXIT_LOOP usually contains IF/END statements, which identify a condition that dictates when a loop should be exited early. In the following example, a REPEAT loop is started that will normally execute 10 times. Halfway through the loop, variable x is checked. If it has a value of 7, the repeat loop is exited and the sequence resumes execution at statement 8:

```
1. REPEAT 10
2.     STATEMENT two
3.     IF x == 7
4.         EXIT_LOOP
5.     END
6.     STATEMENT six
7. END
8. STATEMENT eight
```

NEXT_LOOP Statement

The NEXT_LOOP statement is similar to the EXIT_LOOP statement. However, execution of the looping block is not canceled completely: Only the current iteration of the loop is canceled. In the following example, if y is less than 10 when statement 4 is reached, the sequence will resume execution at statement 1:

```
1. REPEAT 10
2.     STATEMENT two
3.     IF y < 10
4.         NEXT_LOOP
5.     END
6.     STATEMENT six
7. END
8. STATEMENT eight
```

CASE Structure (Includes VALUE and ANY Statements)

The CASE structure is similar to the IF...THEN...ELSE statement except that it can conditionally execute any of several different blocks of code based on the value of a variable. A CASE structure is actually built from the CASE, VALUE, ANY, and END statements. In the following example, if the variable **code** has a value of 1, STATEMENT one will execute. If it has a value of 2, STATEMENT two will execute. If it has any other value, STATEMENT three will execute.

```
1. CASE code OF
2.     VALUE 1:
3.         STATEMENT one
```

```

4.      VALUE 2:
5.      STATEMENT two
6.      ANY
7.      STATEMENT three
8.  END

```

GOTO Statement

The GOTO statement allows you to jump to any other labeled step in a sequence. You create a step “label” by typing any 15-character label on a line displaying the `--statement--` place holder. The label must begin with a character and must not include any spaces. It can contain mixed alpha and numeric characters. The label must be terminated with a colon “:”. In the following example, if the variable **check** has a value of 7, a GOTO will be executed and execution will jump to line 15 of the sequence:

```

15. steplabel:
    .
    .
    .
20. IF check == 7
21.     GOTO steplabel
22. END
23.

```

SET Statement

The SET statement allows you to assign a new value to a variable.

The syntax is:

```

SET --o_variable-- = {--uopr--} --variable-- {--opr--
--variable-- {--opr-- --variable-- {--opr-- --variable--
{--opr-- --variable-- {--opr-- --variable--}}}}

```

The following example would result in **var1** having the value of **var2 + var3 – 12**:

```
SET var1 = var2 + var3 - 12
```

NOTE: Both the SET and SETS instructions use string or numeric values interchangeably. If a numeric is found where a string is expected, a string value is created using the V⁺ /D format (except that the leading space is suppressed). If a string is found where a numeric is expected, the numeric value is found by using the V⁺ VAL() function. In this case, nonnumeric strings result in a value of zero. Refer to the *V⁺ Language User's Guide* for details.

SETS Statement

The SETS statement allows you to assign a new value to a string variable.

The syntax is:

```

SETS --o_variable-- = {--s_uopr--} --variable--
{--s_opr-- --variable-- {--s_opr-- --variable--
{--s_opr-- --variable-- {--s_opr-- --variable-- {--s_opr--
--variable--}}}}

```

The following binary and unary operators (shown in bold type) may be used for the statement argument:

--s_opr--

Operators	Result
a + b	Concatenates strings a and b. For example, "under" + "ground" = "underground".
a LEFT_OF b	Finds the left-most occurrence of b in a. Evaluates to the substring to the left of b. If b is not found, evaluates to an empty string.
a RIGHT_OF b	Finds the left-most occurrence of b in a. Evaluates to the substring to the right of b. If b is not found, evaluates to all of a.
a LEFT_OF_ANY b	Finds the left-most occurrence of any character from b in a. Evaluates to the substring to the left of that character. If no character is found, evaluates to an empty string.
a RIGHT_OF_ANY b	Finds the left-most occurrence of any character from b in a. Evaluates to the sub-string to the right of that character. If no character is found, evaluates to all of a.
a LEFT n	Evaluates to the left-most n characters of a.
a RIGHT n	Evaluates to the right-most n characters of a.
a REMOVE b	Removes any characters in B from the left of a.
a REMOVE_NOT b	Removes any characters not in b from the left of a.
UPPER	Converts all characters in the string to upper case.
LOWER	Converts all characters in the string to lower case.

NOTE: Both the SET and SETS instructions use string or numeric values interchangeably. If a numeric is found where a string is expected, a string value is created using the V⁺ /D format (except that the leading space is suppressed). If a string is found where a numeric is expected, the numeric value is found by using the V⁺ VAL() function. In this case, nonnumeric strings result in a value of zero. Refer to the *V⁺ Language User's Guide* for details.

RETURN Statement

The RETURN statement causes the current sequence to stop executing as if the end of the sequence had been reached and all cycles had been completed. If this sequence was initiated by a CALL statement, the sequence which executed the CALL statement resumes execution at the statement following the CALL. If executed from the main sequence, the runtime task returns to idle state.

The syntax is:

```
RETURN
```

13.4 Sequence Control Statements

The following statements allow you to control execution of your sequences. One of the primary uses of these statements is to load and execute all the sequences that are required for a particular operation. This allows users to start up a complicated workcell by simply choosing a single button.

Examples of control sequences are provided in the module **mowctl**. These sequences are associated with the buttons on the Master Control Panel. The sequences in **mowctl** give you examples of how to create control sequences that can be edited so that the buttons perform your custom operations.

NOTE: You do not have to use these example sequences to use the Master Control Panel buttons.

Sequences with the same names can be created in your own modules. When a module is loaded, the buttons on the Master Control Panel will execute the corresponding sequence. For example, create a module named **new**, which contains the control sequence named **proceed**. Edit the record CONTROL SEQUENCE MODULE in the **mowini** initialization database and change it to **new**. Choosing **Proceed** on the Master Control Panel will run the sequence named **proceed** from the module named **new**.¹

LOAD Statement

This statement loads a resource module and all its components.

The syntax is:

```
LOAD MODULE module
```

PAUSE_TASK Statement

The PAUSE_TASK statement performs the same operations as the Immediate, Operation, Action, Statement, and Cycle check boxes on the control panels.

The syntax is:

```
PAUSE_TASK --task-- {AFTER --event--}
```

Double-click **--task--** to select a task name. (These are the same names that appear in the scrolling window on the Task Control Panel). For an immediate pause, leave the **--event--** argument blank. To select other pauses, double-click **--event--**, and the

¹ You can also edit the definitions of the various buttons to specify which module you want to use for the control sequences (see the *AIM Customizer's Reference Guide*).

remaining pause options are displayed. Pauses after Operation and Action depend on the particular statement that is executing.

RESUME_TASK Statement

This statement restarts a task that has been paused with a PAUSE_TASK. It is equivalent to choosing the proceed/retry buttons on the control panels.

The syntax is:

```
RESUME_TASK --task-- {RESPONSE --response--}
```

Double-click --task-- to select a task name. (These are the same names as appear in the scrolling window on the Task Control Panel.) To select proceed/retry options, double-click --response--, and the options are displayed. If --response-- is left blank, the response is the same as the Proceed button.

SELECT_TASK Statement

The SELECT_TASK statement performs the same function as the Select button on the Task Control Panel. A sequence is assigned to a task for execution. If the task is not found, no error is reported.

The syntax is:

```
SELECT_TASK --task-- {MODULE --module--} {SEQUENCE  
--string--}
```

Double click --task-- to select a task name. (These are the same names as appear in the scrolling window on the Task Control Panel). If --module-- is not specified, the module currently assigned to the task is used. If a module is specified, it must be loaded.

START_TASK Statement

The START_TASK statement begins execution of a task. If the task is not found, no error is reported.

The syntax is:

```
START_TASK --task--
```

STOP_TASK Statement

The STOP_TASK statement stops the specified task.

The syntax is:

```
STOP_TASK --task-- {AFTER --event--}
```

Double-click --task-- to select a task name. (These are the same names as appear in the scrolling window on the Task Control Panel). For an immediate stop, leave the --event-- argument blank. To select other pauses, double click --event-- and the remaining pause options are displayed.

As from the control panel, pauses after Operation and Action depend on the particular statement that is executing.

UNLOAD Statement

The UNLOAD statement removes a loaded module from memory. If you attempt to unload a module that is assigned to some task, a warning message is displayed and the module will not be unloaded.

The syntax is:

```
UNLOAD MODULE --module--
```

WAIT Statement

The WAIT statement causes a sequence to pause for the specified time in hundredths of a second.

The syntax is:

```
WAIT time
```

Click on the “time” argument and type the wait time in hundredths of a second (for example, a value of 500 equals 5 seconds). The time argument is a constant.

WAIT_VAR Statement

The syntax for WAIT_VAR is:

```
WAIT_VAR --variable--
```

WAIT_VAR is similar to WAIT. However, the time is specified in seconds instead of hundredths of a second. In this statement, the time argument may be an AIM variable or a constant. (Note that real constants may be specified.) Additionally, the WAIT_VAR statement initiates a V+ WAIT_EVENT.

MOTIONWARE USER'S GUIDE

PART TWO

**MOTIONWARE
WITH VISION**

MotionWare and Vision

14

14.1	Getting Started With Vision	192
14.2	Camera Calibration	192
	Camera Calibration Records	193
	Camera Calibration: Basic Steps	193
14.3	Calibration Records	194
14.4	Calibration Options	196
	Vision Z-Axis Direction	196
	Fixed-Mount Calibration Option	197
	Fixed-Mount: General Case	197
	Downward-Looking With Vacuum Gripper	199
	Arm-Mounted Camera Calibration	200
14.5	Calibration-Process Window	202
14.6	MotionWare With Vision Statements	203
	INSPECT	203
	PICTURE Statement	204
	Vision Inspection Using a Fixed-Mount Camera	204
	Vision Inspection Using an Arm-Mounted Camera	206
14.7	REFINE_TOOL Statement	207
	Sequence of Operations	207
	Vision Refinement for the Gripped Part	208
14.8	REFINE_FRAME Statement	210
	Using REFINE_FRAME	210
	Statement Syntax and Arguments	211
	Sequence of Operations	211
	Records Required by REFINE_FRAME	211
	Fiducial Location Records	211
	Vision Tool Record	212
	Optional Records for REFINE_FRAME	213
	Additional Considerations	214

14.1 Getting Started With Vision

MotionWare uses the vision option for two primary purposes: inspecting parts and locating parts. This chapter discusses inspecting and locating stationary parts. Chapter 15 discusses using the vision system with conveyor systems.

Before you can use the MotionWare vision option, you must be familiar with the tools and vision operations provided in the vision system. These tools and operations are described in the *VisionWare User's Guide*. All of the vision capabilities that are described in this manual are available to MotionWare. The main difference between performing inspections with MotionWare and with VisionWare is that VisionWare groups and optimizes vision operations for speed. MotionWare adds a vision server task that optimizes overlap of vision and motion tasks. Additionally, MotionWare adds several statements for locating parts and their assemblies in the robot coordinate system.

NOTE: Before you continue, read Chapters 2 - 6 in the *VisionWare User's Guide* to learn what types of inspections can be made. Read Appendixes C - F in that guide to learn how to use the tools that the vision system provides for making inspections.

14.2 Camera Calibration

MotionWare includes a predefined global camera database that contains definitions for standard camera calibrations. It also includes a resource module named **vcalmow**, which contains sample vision calibration records that reference the global camera database.

You can use these standard databases in several different ways:

Load and select resource module **vcalmow** and use it to perform your camera calibrations and store the data in the global camera database. Omit the camera database from your custom resource modules so that you will access the global one.

Import the global camera database (named VCAMMOW.DB) and the sample vision database (named VCALMOW.VI) into your custom resource module. The camera calibration will affect only that module.

Copy and Paste individual records from the global camera database and the **vcalmow** vision database to your custom databases.

To better understand how camera calibration works, or to define your own calibrations from the beginning, read the sections that follow.

In order to use the vision system to locate parts for MotionWare, the system must know several things about your cameras and their setup.

This information includes:

- How the camera is mounted.

Fixed-mount camera:

- facing up (in relation to the robot coordinate system)
- facing down
- at an angle

Robot-mounted camera:

- The robot joint where the camera is mounted

- The relationship between the camera field of view and the robot end of arm tooling. (This relationship is calculated using built-in MotionWare calibration utilities.)
- The type of camera being used
- Robot capabilities

Camera Calibration Records

Camera calibration requires a minimum of four records. These four records specify all of the information that is needed by MotionWare in order to calibrate your cameras and make them ready for inspection and guidance operations.

The four records are:

Camera Calibration Record

This record describes how the camera is to be calibrated.

Calibration Object Locator Record

This record finds the calibration object in the field of view. Any vision record that returns a point result can be used as the object locator. See the *VisionWare User's Guide* for details on point-class records. This record is described in this chapter.

Picture Record

This record defines the parameters to be used when a new picture is taken. See the *VisionWare User's Guide* for details on this type of record.

Camera Record

This record maintains the information for a specific camera and receives the calibration data. See the *VisionWare User's Guide* for details on this type of record.

NOTE: The picture record, the camera calibration record, and the calibration object locator are stored in the Vision database. Therefore, before you can calibrate a camera, a resource module must be selected that includes a Vision database. The camera record can be from either the global Camera database or the selected resource module's Camera database (that is, the module that includes the previously mentioned Vision database).

Camera Calibration: Basic Steps

There are many different types of cameras and options for camera calibration. The following discussion covers the basic steps that are common to all calibration operations.

1. Create a camera record. This record tells MotionWare which physical camera to use, the type of camera being used, the robots or conveyors associated with the camera, and other information regarding the camera use. See the *VisionWare User's Guide* for details on camera records.
2. Create a picture record that uses the camera record created in step 1. See the *VisionWare User's Guide* for details on picture records.
3. Create the calibration object finder record. Use the picture record created in step 2 as the picture source record. The calibration-object finder record can be any vision record that belongs to the point class. The actual record used will depend on your calibration method and the calibration object that you use.

For example, for **Fixed mount - general case**, the calibration object is normally a round disk with a beveled hole in the center. During calibration the object is placed in four different locations and the object finder record must be able to locate the object center in all four locations. A Blob finder record works well for this operation. An arc finder record also would work well. See the description of point class records in the *VisionWare User's Guide* for details on the different point-class records.

4. Create a calibration record. This process is described in the next section.

14.3 Calibration Records

To create a calibration record:

Edit ⇒ Vision ⇒ Edit ⇒ New ⇒ Calibration Record Types ⇒
enter record name ⇒ select camera mounting ⇒ OK

The following screen is displayed:

The screenshot shows the 'Adept-Link2-mounted Calib (vcalmow)' window. It has a menu bar with 'Go', 'Seek', 'Edit', and 'Help'. The main area is divided into several sections:

- Top Section:** Contains a text field 'adept_mnt_cal' (labeled 1), a date/time field '18-Feb-95 17:04', and a 'vision system:' dropdown set to '1'.
- Object Locator Section:** Includes 'OBJECT LOCATER name' (adept_mnt_blob), 'PICTURE name:' (adept_mnt_pic, labeled 2), and 'CAMERA name:' (adept_mnt_cam). Each has a 'Goto' button.
- Device Section:** 'Device:' is set to 'Robot_1' (labeled 3). A 'Nominal Location' button is next to it.
- Procedure Specifications Section (labeled 4):** Contains four radio buttons: 'Will touch part with pointer or gripper' (selected), 'Use known dot location', 'Rough calibration w/o touching part', and 'Lefty-Righty calib, w/o touching part'. Below are 'Touch mechanism:' options: 'Pointer' (selected) and 'Vacuum Gripper'. A 'Setup Details' button is present.
- Camera Specifications Section (labeled 5):** Includes 'Physical camera:' (1), 'Camera model #:' (0), 'Mounted on link:' (2), 'Full frame' (selected), 'Field only (half-rez)', 'Shutter Use' (Synchronous and Asynchronous), and 'Use perspective correction' (unchecked).
- Buttons:** 'Set Cal to Default', 'Undo Last Calib' (labeled 6), and 'Calibrate/Test...'.
- Calibration Results Section (labeled 7):** Includes 'Xscale:' (1.0000), 'Yscale:' (1.0000), 'X/Y ratio:' (1.0000), 'Max error (w/o persp. corr.):', and 'To.cam:'.

Figure 14-1
 Vision Calibration Window

Refer to Figure 14-1

- ❶ Shows the vision system that will be used by this calibration record. The vision system is specified in the camera record associated with the object locator (item ❷).
- ❷ Select the object finder record that you are going to use for this calibration. The object finder record must come from the point class of records and be able to locate the calibration object wherever it may be positioned in the field of view.

See the description of point-class records in the *VisionWare User's Guide*. See the description of the various calibration options in the following sections. There is a wide variety of records that might be used to locate the calibration object. The record selected will depend on your particular application.

When you select a calibration object, the picture and camera records associated with the calibration object record is displayed. Choose **Goto** to display the picture or camera records. Ensure that these records specify the options that correspond to your camera and robot setup.

- ❸ Shows the robot that is associated with this camera (selected in the camera record). Choose **Nominal Location** to define a location to which the robot may move that is clear of the field of view. During calibration the robot will move to this location whenever a new picture is needed (fixed-mount cameras).
- ❹ This area will vary depending on the particular calibration method you choose. The various calibration methods are described in the following sections.
- ❺ Shows the camera definition. All information in this area is specified in the camera record. Choose **Goto**, in item ❷, to display the camera record associated with this calibration record.
- ❻ Choose **Set Cal to Default** to return all calibration data to its default status.
Choose **Undo Last Calib** to return all calibration data to its previous values.
Choose **Calibrate/Test** to begin the calibration process (described below).
- ❼ This area shows the calibration data:
 - X-scale pixel per mm in X direction
 - Y-scale pixel per mm in Y direction
 - X-Y ratio of X pixel to Y pixel
 - Max error shows maximum error that will be obtained from the system if perspective calibration is not used
 - To.cam shows the actual transformation that relates the camera field of view to the robot world coordinate system

14.4 Calibration Options

There are four primary calibration options:

Arm-mounted

This includes any camera that is mounted on a robot's moving link. This calibration allows the robot to move to arbitrary locations and acquire images for inspection or guidance.

Fixed-mount

This includes all cameras that are mounted at a fixed location within the robot working envelope. Cameras may be mounted facing up or down relative to the robot world Z-axis.

Conveyor-mount

This includes cameras that will locate parts on conveyor belts. This camera must view the desired working width of the belt. The camera must be upstream of the robot work envelope, but it does not have to be within the work envelope.

Camera only

This includes cameras that will be used for inspection only. Cameras mounted on a robot (moving to the same location) for inspecting parts can use camera-only calibration. (The camera calibration will be specific to a given camera to work surface distance and can be used only at that distance.) This type of calibration is detailed in the *VisionWare User's Guide*.

* Important points about camera calibration:

- The millimeter-to-pixel ratio will be different for different camera-to-object distances. If you are making distance measurements of objects that are different distances from the camera, you will have to create a calibration data set (virtual camera) for each distance.
- When you change the focus of a camera lens, the amount of the viewing area in the field of view changes. This means that the millimeter-to-pixel ratio will change. Camera calibrations created before the lens was refocused will not be valid, and the camera will have to be recalibrated.
- A prototype model record and all prototype finder records that locate the prototype model must use the same virtual camera (see the *VisionWare User's Guide*).
- If you move a camera, the camera-to-robot base distance will no longer be correct, and the camera will have to be recalibrated.

Vision Z-Axis Direction

When the vision system loads the camera calibration data, it assumes that the Z-axis of the robot transformation is pointing toward the camera. In some cases, this is not the correct orientation and a change in the camera database will have to be made. One case where this occurs in MotionWare is using an upward-mounted camera to locate objects on a translucent conveyor belt. Other instances requiring this change are detailed in the next two chapters.

To change the Z-axis direction:

Choose for camera name, select ☒ **Points toward camera**, or ☐ **Points away from camera**.

Fixed-Mount Calibration Option

The following sections describe the options for calibrating fixed-mount cameras.

Fixed-Mount: General Case

Open a fixed-mount camera calibration record, select ☒ **Fixed-mount - general case**, and choose **Setup Details**.

The following screen is displayed:

Fixed-mount Calibration (vcalmow)

Go Seek Edit Help

fixed_mnt_cal Vision system: 1
18-Feb-95 17:05

OBJECT LOCATER name: fixed_mnt_blob

PICTURE name: fixed_mnt_pic **Goto**

CAMERA name: fixed_mnt_cam **Goto**

Device: Robot_1 **Away Location**

1 Procedure Specifications

- ☐ Fixed-mount - general case
- ☒ Downward-looking - with vacuum gripper
- ☐ Fixed-mount - object on belt, using ptr

Touch mechanism:

2 ☒ Pointer **Setup Details**

☐ Vacuum Gripper

Camera Specifications

Physical camera: 1 Camera model #: 0

☐ Full frame

☒ Field only

☒ Use

Set Cal to D

Undo Last C

Cal

Xscale: 1.0

Yscale: 1.0

Max error (w/o

To.cam:

Setup Details

Procedure Parameters

3 ☒ Camera is mounted roughly vertically in the world coordinate system

4 ☒ All object locations should roughly have the same world Z value

Touch Mechanism Specifications

5 ☒ Robot's tool can rotate about its Z axis

Number of rotated positions to use: 4

☒ Pointer On signal: 3001 ☐ **Set Z-ht...**

☐ Vacuum Gripper **6** Off signal: 0 ☐

Robot Motion Parameters

7 Overall robot speed: 80 ☐ SINGLE mode

Robot depart distance: 50 **8** ☒ MULTIPLE mode

9 Robot settling time before each picture: 0.5

OK

Figure 14-2
Fixed-Mount Camera Calibration

The general case has two options for calibrating. In the first option, a pointer is mounted to the robot tool flange. You will be prompted to place the pointer in the center of the disk. You will then be prompted to lower the pointer to the center of the disk using only Z-motion. During the Z-only motions, you should not move the pointer but allow the disk to center under the pointer.

The pointer should have a ball end and the disk should have a beveled hole in the center to allow this self-centering. You will be prompted to place the disk in each of the four quadrants of the field of view. The calibration object finder record must be able to find the disk in all four quadrants. Thus, the tool search area must encompass most of the field of view.

The second option uses a vacuum gripper attached to the tool flange and a disk that the vacuum gripper can pick up. The robot will pick up the disk several times and move it around the field of view. The gripper should positively release the disk without moving it.

Downward-Looking With Vacuum Gripper

Open a fixed-mount camera calibration record and select **Downward-looking - with vacuum gripper**. This operation will be similar to the general case, except that the pointer option will not be available.

Once you have completed the set-up options listed below, you can choose **Calibrate/Test** to begin the calibration process. The window shown in Figure 14-4 is displayed.

- ❶ Select the desired calibration option. The **Downward-looking - with vacuum gripper** is a special case of the **Fixed-mount - general case**.
- ❷ Indicate whether you are using a pointer or vacuum gripper. Choose **Setup Details** to display options ❸ through ❹.
- ❸ Select this option if the camera is mounted roughly parallel to the robot Z-axis.
- ❹ Select this option if all the parts you will acquire will be in approximately the same X/Y plane.
- ❺ Select this option if the robot has an axis that can rotate about the robot Z-axis (the fourth axis of a SCARA robot, for example). Indicate how many times you want the robot to pick up or center on the calibration disk. Four is a good number unless you find that you are not getting sufficient accuracy from your calibration.
- ❻ If you have selected the vacuum gripper option, indicate the output signals that control the gripper.
- ❼ Specify the robot speed for motions made during calibration and the distance the robot should retract Joint 4 (Z-axis) before moving to the next location.
- ❽ In robots that can make more than one rotation about the Z-axis, specify whether multiple turns are allowed. (This option allows you to prevent vacuum or electrical lines going to the end effector from becoming wrapped around the Z-axis.)
- ❾ Indicate how much time the robot should settle before taking each picture. This helps eliminate any camera vibration that may occur after a motion.

Arm-Mounted Camera Calibration

Select ☒ **Will touch part with pointer or gripper** and choose **Setup Details** to display the following screens:

The screenshot shows the 'Arm-Mounted Camera Calibration' dialog box. It is divided into several sections. Callout 1 points to the 'Procedure Specifications' section where the first option, 'Will touch part with pointer or gripper', is selected. Callout 2 points to the 'Touch mechanism' section where 'Pointer' is selected. Callout 3 points to the 'Camera Specifications' section, specifically the 'Physical camera' and 'Mounted on link' fields. Callout 4 points to the 'Setup Details' sub-dialog box, specifically to the 'Procedure Parameters' section. Callout 5 points to a checkbox in the 'Procedure Parameters' section. Callout 6 points to another checkbox in the 'Procedure Parameters' section. Callout 7 points to the 'Touch Mechanism Specifications' section where a checkbox is selected. Callout 8 points to the 'Touch Mechanism Specifications' section where 'Vacuum Gripper' is selected. Callout 9 points to the 'Robot Motion Parameters' section, specifically the 'Overall robot speed' field. Callout 10 points to the 'Robot settling time before each picture' field.

Procedure Specifications

- ☒ Will touch part with pointer or gripper
- ☐ Use known dot location
- ☐ Use rotation of camera, no touching part

Touch mechanism:

- ☐ Pointer
- ☒ Vacuum Gripper

Camera Specifications

Physical camera: Camera model #:

Mounted on link:

☒ Full frame ☐ Field only (half-rez)

☐ Synchronous ☐ Asynchronous

☒ Use perspective correction

Setup Details

Procedure Parameters

- ☒ Camera is mounted roughly vertically in the world coordinate system
- ☐ All object locations should roughly have the same world Z value
- ☒ Camera is pointed in roughly the same direction as the Z-axis of the link that the camera is mounted on

Touch Mechanism Specifications

- ☒ Robot's tool can rotate about its Z axis

Number of rotated positions to use:

Robot Motion Parameters

Overall robot speed: ☐ SINGLE mode ☒ MULTIPLE mode

Robot depart distance:

Robot settling time before each picture:

OK

Figure 14-3
Arm-Mounted Camera Calibration

- 1 Select the desired calibration option.

The ☒ **Will touch part with pointer or gripper** option requires that you are able to place the pointer or gripper at the center of the calibration object.

The **● Use known dot location** requires that you accurately know the location of a point in the work envelope.

The **● Use rotation of camera -no touching part** option requires that the camera is able to move to several locations around the calibration object.

- 2 Indicate whether you are using a pointer or vacuum gripper. Choose **Setup Details** to display options 4 through 10.
- 3 Indicate on which robot link the camera is mounted. The link number must correspond to the kinematic description provided in the device module documentation for your mechanism. The link numbers for Adept robots are provided in the robot user's guide.
- 4 Select this option if the camera is mounted roughly parallel to the robot Z-axis.
- 5 Select this option if all the parts you will acquire will be in approximately the same X/Y plane.
- 6 Select this option if the camera is pointed down in relation to the robot world coordinate system. See the device module documentation of the robot user's guide.
- 7 Select this option if the robot has an axis that can rotate about the robot Z-axis (the fourth axis of a SCARA robot, for example). Indicate how many times you want the robot to pick up or center on the calibration disk. Four is a good number unless you find that you are not getting sufficient accuracy from your calibration.
- 8 If you have selected the vacuum gripper option, indicate the output signals that control the gripper.
- 9 Specify the robot speed for motions made during calibration and the distance the robot should retract along the Z-axis before moving to the next location.

In robots that can make more than one rotation about the Z-axis, specify whether multiple turns are allowed. (This option allows you to prevent vacuum or electrical lines going to the end effector from becoming wrapped around the Z-axis.)
- 10 Indicate how much time the robot should settle before taking each picture. This helps eliminate any camera vibration that may occur after a motion.

During calibration using the **● Will touch part with pointer or gripper** option, you will be asked to center the pointer or vacuum gripper on the calibration object. The system will pick up and release the object (or recenter the object) several times, rotating the gripper (if possible) each time. Once you start the calibration process, directions for completing the calibration will be shown on the screen (see the next section).

14.5 Calibration-Process Window

When you begin calibration, the following window is displayed. It shows the selected camera options and current calibration status. Messages are displayed that will tell you what the calibration routines are doing and what you need to do during different phases of calibration.

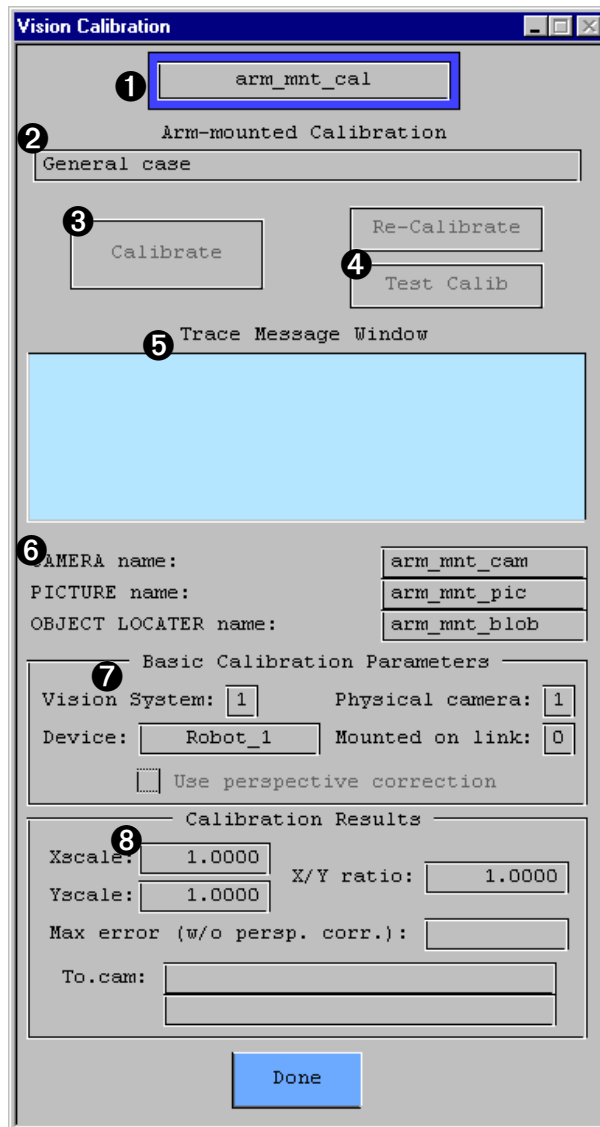


Figure 14-4
Calibration-Process Window

- 1** Displays the name of the calibration record.
- 2** Displays the selected calibration method.
- 3** Choose to begin the calibration process.
- 4** After completing a calibration session, if the results are not satisfactory, choose **Re-Calibrate** to begin a new calibration session.

Choose **Test Calib** to verify the accuracy of a calibration.

- ⑤ This area shows the status of the current calibration session.
- ⑥ This area shows the source records for the calibration record.
- ⑦ This area shows important selections from the camera record.
- ⑧ This area shows the calibration results.

14.6 MotionWare With Vision Statements

The following statements are detailed in this section: INSPECT, PICTURE, REFINE_FRAME, and REFINE_TOOL. The statement SET_FRAME, detailed in Chapter 13, also has an optional vision argument.

INSPECT

The INSPECT statement causes evaluation of up to eight inspection records. Records are evaluated until the first inspection fails, at which time execution stops and the OK_SIGNAL is set to FALSE. If all inspections pass, then the OK_SIGNAL is set to TRUE.

The syntax is:

```
INSPECT OPERATION --vision-- { --vision-- { --vision-- } { _
    --vision-- { --vision-- { --vision-- { --vision-- { _
    --vision-- } } } } } {SAVE_PREVIOUS --yes/no--} {OK_SIGNAL
    --o_variable--}
```

The `--vision--` argument can be completed only with an Inspection record. An inspection record will appear in the pick-list of available records only if ☒ **TopLevel** is selected in that record.

The one required argument for the INSPECT statement is the first `--vision--` argument (no brackets around the argument indicate it is a required argument). When this statement is executed, vision will take all the necessary pictures and evaluate all the necessary records to generate the value that the inspection record is to evaluate. (This statement will move the robot as necessary to take pictures, depending on the camera mounting and the parameters specified in the picture records.) The inspections are evaluated and the output variable, if defined, is set to TRUE (-1) if all the inspections succeed. Otherwise, it is set to FALSE (0). If ☒ **Gather stats** was selected in the inspection record, the values generated by the inspection are reported to the Inspection Results pages (see *VisionWare User's Guide*).

The optional arguments (`{--vision--}`) also require a top-level inspection record. Each of these records will be evaluated in the same manner as the required inspection record, with two exceptions. First, inspection records are evaluated left-to-right until an inspection fails. After an inspection fails, no more inspection records will be evaluated. Second, if an inspection fails, the entire statement will be considered to have failed, and the variable specified in `{OK_SIGNAL - o_variable - }` will be set accordingly.

If `{SAVE_PREVIOUS}` is omitted or set to "no", all pictures needed for the inspection are taken. If a vision record that requires movement of the robot is in the inspection tree, the robot will move. If this argument is set to "yes", the system assumes that all picture data is available and will not acquire new images (although it will wait for any in-progress image

processing to complete). If any required image data is not available, the sequence will halt with the appropriate error.

PICTURE Statement

The PICTURE statement takes the pictures required for a vision operation and (if specified) moves the robot to a picture-taking location.

The syntax is:

```
PICTURE FOR --vision--
    {SINGLE_PICTURE --vision--{SAVE_PREVIOUS --yes/no--}}
    {OK_SIGNAL --o_variable--}
```

This statement takes pictures for the vision operation specified in the `FOR --vision--` clause and the vision server begins computing the vision result. This statement does not wait for the vision result to be computed but completes as soon as all the requested pictures are taken.

The robot may be moved if the picture records specify robot motions.

If the `SINGLE_PICTURE --vision--` clause is not specified, all pictures for the entire vision operation are taken.

The `SINGLE_PICTURE` clause is useful only in situations where the vision operation is based on more than one picture. If the `SINGLE_PICTURE --vision--` clause is specified, its argument must be the name of a picture record in the vision database. This statement then takes only that picture and completes as soon as the picture is taken. In the background, the vision server computes any vision results that depend on that picture.

If the `SAVE_PREVIOUS --yes/no--` clause is omitted or set to *no*, all previous vision data for the initial vision operation is cleared and recomputed.

If the `SAVE_PREVIOUS --yes/no--` clause is set to *yes*, the vision data that depends on this picture is cleared, but the remaining data for the vision operation is not cleared. This setting is used after the first picture in a multiple-picture application.

If the `OK_SIGNAL --o_variable--` clause is specified, the output variable is set to TRUE if the picture taking succeeded. Otherwise it is set to FALSE. This variable does not reflect the status of the entire vision operation.

See the *VisionWare User's Guide* for details on creating vision records.

Vision Inspection Using a Fixed-Mount Camera

To specify the motion information for a picture-taking location that is based on a fixed-mount camera:

Edit ➡ Vision ➡ Seek ➡ Index ➡ *double-click the picture record*

1. If you want the robot to move as a part of the picture-taking operation, select ☒ **Use it** from the Motion group and choose **Setup**.¹
2. When the motion information pop-up window is displayed, select ☒ **Object moved by robot** and the following options are displayed:

¹ If options are not available, they are not appropriate to the current camera calibration.

Figure 14-5
Picture Record Motion Options (Fixed-Mount Camera)

- ❶ Select this option if the robot is holding the object in front of the camera when the picture is taken.
- ❷ Select ☒ **Robot location** to move the robot to the path point specified by items ❸ and ❺ when taking a picture.

Select ☒ **Vision target location** and the robot will use the robot-to-field-of-view transformation created during camera calibration to move the robot gripper over the camera when taking a picture. The transformation in the path point specified by items ❸ and ❺ will be interpreted as an offset from the center of the corner of the field of view, rather than an absolute robot location.

- ❸ Picture-taking locations are taken from the path database. Double-click this data box to display a list of defined paths.

If ☒ **Robot location** is specified, the path segment will be the actual robot location.

If ☒ **Vision target location** is specified, the path segment represents an offset from the vision target location. The path segment used for the picture-taking location is specified in item ❺.

- ❹ To ensure all robot motion and camera vibration has stopped, it may be necessary to add settling time to the end of the robot motion. Small fields of view, heavy payloads, and high robot speeds require longer settling times. Use 0.3 as a starting point.
- ❺ Enter the path segment number to use as the location. If ☒ **Auto-select** is used, segment 1 will be selected.
- ❻ If ☒ **Vision target location** is selected, values in these data boxes will cause the robot to offset the center of the camera field of view from the location. The default values (50/50) center the robot at the vision location. This offset is in addition to any offsets defined in the path location.
- ❼ Choose **Here** to record the current location in the specified path database record as the picture-taking location.

If ☒ **Robot location** is selected, the position of the robot tool tip is recorded.

If ☒ **Vision target location** is selected, the position of the object being viewed by the camera is recorded.

- 8 Choose **Done** to close this pop-up window.

Vision Inspection Using an Arm-Mounted Camera

To specify the motion information for an arm-mounted camera:

Edit ➔ **Vision** ➔ **Seek** ➔ **Index** ➔ *double click the picture record*

If you want the robot to move as a part of the picture-taking operation, select ☒ **Use it** from the Motion group and choose **Setup**.¹

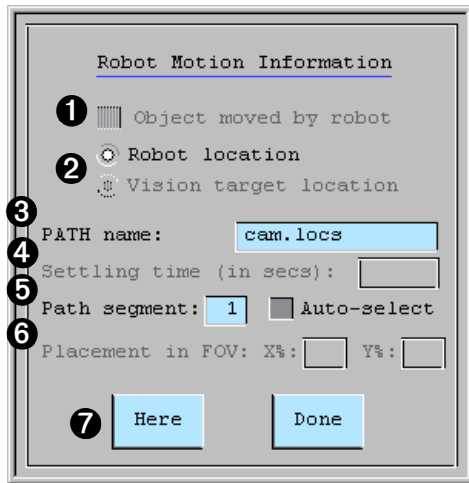


Figure 14-6
Picture Record Motion Options (Arm-Mounted Camera)

- 1 Normally, since the robot is moving the camera over a stationary part, you will not select ☒ **Object moved by robot**. You would select this option in the special case of a camera mounted on a robot link such that the gripped part could be viewed by the camera.
- 2 Select ☒ **Robot location** to move the robot to the location specified in item 3 when taking a picture.

Select ☒ **Vision target location** and the robot will use the robot flange-to-field-of-view transformation created during camera calibration to move the camera (rather than the gripper) over the picture-taking location.

If you are using a computed frame for the vision inspection, and the tools needed to compute the frame will not fit on a single screen, you can use multiple picture records with different path segments to move the camera to the different locations. MotionWare will take care of the necessary calculations to generate the frame.

- 3 Picture-taking locations are taken from the path database. Double-click this data box to display a list of defined paths. Leave this argument blank to prevent the robot from moving (for example, if you have already positioned it with a MOVE statement).

¹ If the "Motion Information" button is not displayed, the camera is not correctly calibrated.

If ☐ **Robot location** is specified, the path segment will be the actual robot location.

If ☐ **Vision target location** is specified, the path segment represents the location of the object to be viewed. The path segment used for the picture-taking location is specified in item **5**.

- 4** To ensure all robot motion and camera vibration has stopped, it may be necessary to add settling time to the end of the robot motion. Small fields of view, heavy payloads, and high robot speeds require longer settling times. Use 0.3 as a starting point.

- 5** Enter the path segment number to use as the location. If ☒ **Auto-select** is used, segment 1 will be selected.

- 6** If ☐ **Vision target location** is selected, values in these data boxes will cause the robot to offset the center of the camera field of view from the location. The default values (50/50) center the robot at the vision location. This offset is in addition to any offsets defined in the path location.

- 7** Choose to record the current location in the specified path database record as the picture-taking location.

If ☐ **Robot location** is selected, the position of the robot tool tip is recorded.

If ☐ **Vision target location** is selected, the position of the object being viewed by the camera is recorded.

- 8** Choose to close this window.

14.7 REFINE_TOOL Statement

The REFINE_TOOL statement uses vision to create or refine a tool offset. The REFINE_TOOL statement moves the robot gripper in front of an upward facing camera, takes a picture, and calculates a vision reference frame for the gripped part. The difference between the part reference frame center and the true gripper center is taken into account when the part is placed. This statement improves part placement accuracy for parts that are not gripped perfectly.

The syntax is:

```
REFINE_TOOL {APPROACH --path--} REFINE --vision--
{AND --vision--} {WAIT --yes/no--} {SAVE_TOOL --tool--}
{{ALONG --path--} MOVE_TO --location-- {MOVE_ONCE --yes/no--}
{DEPART --path--}{USING --tool--}} {OK_SIGNAL --o_variable--}
```

Sequence of Operations

The sequence of operations performed by this statement is as follows (the relevant portion of the statement syntax is shown for each step):

- If defined, set the optional tool ({USING --tool}).
- If defined, enter the approach path ({APPROACH --path--}).
- Move to the picture-taking location¹ specified in the picture record associated with the vision record. Perform the vision operation (REFINE --vision--).

¹ A vision frame may use more than one camera to create the frame. If this is the case, the robot will move to each required picture-taking location.

- If defined, rotate the gripper 180° and perform a second vision operation. This operation further compensates for inaccuracies in the camera-to-robot calibration ({AND --vision--}).
- Calculate the tool offset.
- If specified, wait for all vision operations to complete before moving. ({WAIT --yes/no--}).
- If WAIT is “no” and a MOVE_TO location is specified, enter the {ALONG --path--} (if defined) and move to the approach segment of the MOVE_TO location (without the tool refinement).
- Wait for the vision operation to complete. If it fails, exit with the optional OK_SIGNAL set to FALSE.
- If specified, store the value in the tool record {SAVE_TOOL --tool--}. Set the current tool to the calculated value. If the vision operation fails, set the tool to undefined.
- If a MOVE_TO location is specified, WAIT is *no*, and MOVE_ONCE is *yes*, do not move to the approach location with the refined tool since the move has already been made with the unrefined tool. Otherwise, if the ALONG path was not used, follow it now and then move to the approach location with the refined tool.
- Move to the MOVE_TO location using the tool refinement.
- If defined, enter the optional depart path {DEPART --path--}.
- If defined, set the output variable {OK_SIGNAL --o_variable--} to TRUE.

The requirements for the vision records used by this statement are described in the next section.

Vision Refinement for the Gripped Part

The REFINES --vision-- and AND --vision-- clauses of the REFINES_TOOL statement require a vision record that returns a vision frame defining the actual location of a part in the robot gripper. The center and orientation of this frame are used to calculate the true part location relative to the center of the gripper. If both vision clauses are used, the results of both operations are averaged to calculate the true part location.

The offset of the true part location from the actual gripper center will be used to compute a new tool transformation. The most commonly used vision tools that return a frame result are:¹ blob finders, prototype finders, and computed frames. See the VisionWare User's Guide for details.

The camera record for the refinement operation should have the following characteristics:

The camera used to generate the frame should be a stationary camera that the robot presents the part to.

The camera should have been calibrated with one of the fixed-mount calibration options (see section 14.2).

The picture record for the refinement operations should have the following characteristics:

In the Robot Motion Information window, select ☒ **Object moved by robot.**

¹ Vision frames are based on the camera coordinate system, not the robot base, and are not interchangeable with reference frames from the frame database.

If ☒ **Vision target location** is selected, specify a path name in the Location data box. The path record controls the motion parameters for moving to the picture-taking location. These include, speed, motion type, etc. The robot-to-camera transformation created during camera calibration will be used as the basic picture-taking location. The location values in the path record are interpreted as an offset from this location. (The path reference frame must be set to ☒ **World**.)

When the robot moves the gripped part in front of the camera, it uses the location values created during camera calibration to move the gripper in front of the camera. The values in the path segment offset the camera from this location.

If you are using a computed frame to calculate the true part center, and the tools needed to compute the frame will not fit on a single screen, you can use multiple picture records with different path segments to move the camera to the different locations. MotionWare will take care of the necessary calculations to generate the frame.

Entering values in the Placement in FOV data boxes will offset the picture-taking location by the specified amount. The default values (50/50) put the center of the gripper in the center of the field of view.

Choose to record the current robot offsets from the camera location in the specified path segment.

If ☒ **Robot location** is selected:

The path segment specified in the Location data box will be the location the robot moves to when taking a picture.

Choose to record the current robot location in the specified path segment.

If ☒ **Auto-select** is selected:

The first vision refinement operation (`REFINE --vision--`) will use segment 1 of the specified path, and the second refinement argument (`AND --vision--`) will use segment 2 of the specified path. If you enter a value in the Path segment data box, that segment will be used during both refine operations.

Things to Remember

1. If your robot is using a tool transformation, teach all locations with that tool transformation as the currently selected tool (use **Setup ➡ Display/Change Tool** to verify that the correct tool is being used). The cameras used for the `REFINE --vision--` and `AND --vision--` clauses must also be calibrated with the tool transformation in place.
2. The values generated by the vision operations are added to any existing values of the current TOOL to create the final placement location. When debugging the `REFINE --vision--` and `AND --vision--` clauses, observe the robot TOOL transformation before and after the refinement. If it changes drastically, either the camera calibration is incorrect or parameters on the picture record or associated path are set up incorrectly.

The current values of TOOL can be seen by:

Setup ➡ Display/Change Tool

14.8 REFINE_FRAME Statement

The REFINE_FRAME statement is used to create a reference frame for an assembly. The reference frame is based on engineering or CAD data for the assembly. The vision system locates fiducial marks on the assembly and uses the known values of the fiducial offsets from the edges¹ of the assembly to create or update an assembly reference frame.

Figure 14-7 shows a typical assembly (a printed circuit board). The exact locations of the two fiducial marks (relative to a corner of the PC board) are known from the board engineering data. MotionWare uses this information to create or update the assembly reference frame for the PC board.

All locations on the board are defined relative to this database reference frame. Each time a new board enters the workcell, the vision system locates the fiducials on the board and updates the database reference frame based on the location of the fiducials. The robot places parts at locations that are defined relative to this updated database reference frame.

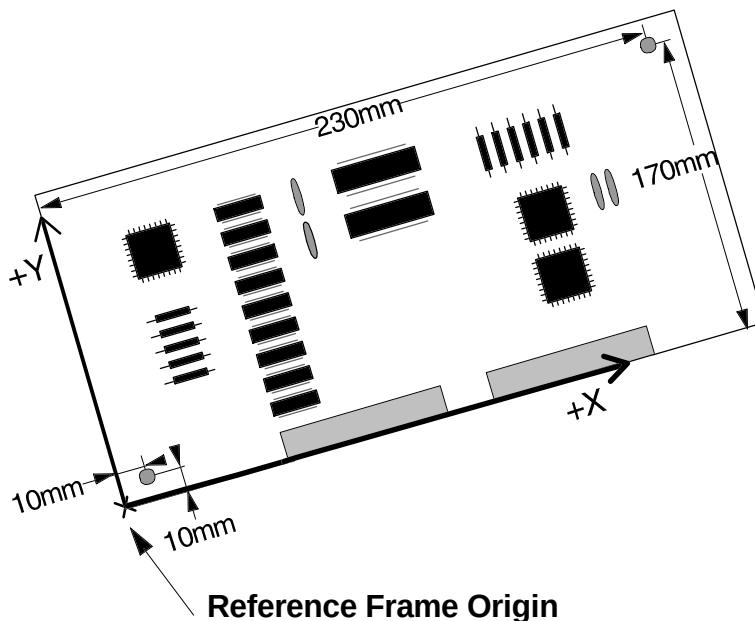


Figure 14-7
REFINE_FRAME Example

In Figure 14-7, the fiducials (located 10 x 10 mm and 230 x 170 mm from the frame origin) are used to calculate the frame origin and orientation relative to the robot workspace.

Using REFINE_FRAME

This statement uses vision to determine the precise value of the reference frame associated with an entire assembly (the assembly database reference frame). This statement visually locates 1 to 4 fiducial marks on the assembly and uses them to correct the nominal value of the reference frame.

¹ Or other features or marks that are the zero-basis data for the location values.

In order to position a camera to view each fiducial mark, an initial approximate reference frame may be specified. In addition, this statement allows optional paths to be followed while moving to the first fiducial mark and departing from the last fiducial mark.

Statement Syntax and Arguments

The syntax for this statement is as follows (braces ({...}) enclose optional clauses):

```
REFINE_FRAME --frame-- {DEFAULT--frame--} {APPROACH--path--}
LOCATE--vision-- {AT--location-- {--location-- {--location--
{--location--}}}} {DEPART--path--} {USING--tool--} {OK_SIGNAL
--o_variable--}
```

Sequence of Operations

The sequence of operations performed by this statement is as follows. The relevant portion of the statement syntax is shown for each step.

1. If defined, apply the tool to describe the current robot gripper. {USING--tool--}
2. If defined, move along the approach path toward the first picture-taking location. {APPROACH--path--}
3. If defined, set the value of the first frame to the value of the default frame. The default frame value must be sufficiently accurate to ensure that the fiducial marks lie within the vision field of view. {DEFAULT--frame--}
4. Move to each picture-taking location and perform the specified vision operation to visually determine the fiducial mark location. LOCATE vision {AT--location-- {--location-- {--location-- {--location--}}}}
5. Compute the actual assembly reference frame, based on the visually determined fiducial mark locations, and set the first frame to this computed value.
6. If defined, depart from the last picture location along the depart path. {DEPART--path--}
7. If defined, set the OK_SIGNAL based on the success of creating a reference frame.

Records Required by REFINE_FRAME

The record names in this section are required for the basic refine frame operation.

Fiducial Location Records

The fiducial locations must be locations in the location database. Creating locations is covered in Chapter 5. REFINE_FRAME requires at least one fiducial mark, but can use up to four fiducials to generate the reference frame. To update accurately the reference frame, these locations should specify the actual engineering values of the fiducial marks (see Figure 14-7).

When creating these locations, remember:

Fiducial locations are actual robot locations, not picture-taking locations. When the robot moves to a fiducial location during a REFINE_FRAME statement, the required offsets will be automatically added by MotionWare. Thus, the camera, rather than the gripper, is positioned over the fiducial location.

● **Named** should be selected as the reference frame for all locations (including the fiducial locations). MotionWare will use the verified fiducial location values

generated by the vision system to update the specified reference frame each time a REFINE_FRAME statement is executed.

If a default frame is not used, the initial database reference frame values must be close enough to the actual assembly location to enable the vision system to find the fiducial marks. Note that without a default frame, the picture-taking locations for the next REFINE_FRAME statement will be relative to the reference frame updated in the last REFINE_FRAME statement.

If a default frame is used, it must be close enough to the location to enable the vision system to find the fiducials (the database reference frame can be left blank). When the default frame is used, the picture-taking locations will always be the same.

Vision Tool Record

A vision record that returns a frame result¹ is required to locate the fiducial in the camera field of view. While any vision tool from the frame class can be used, the vision tool records most commonly used to locate fiducials are:

- A blob-finder record (described in Appendix D of the *VisionWare User's Guide*)
- A prototype model record and a prototype-finder record (described in Appendixes C and D of the *VisionWare User's Guide*)
- A computed frame record (described in Appendix E of the *VisionWare User's Guide*)

A blob-finder record is the simplest vision record to use. It merely locates a bounded region within the tool search area and returns the orientation and center of the area. If the fiducial marks on an assembly can be uniquely identified using blob finders, they provide the simplest method of locating fiducials.

If only one fiducial mark is used, the orientation of the frame is calculated from the orientation of the found blob. This requires that an ellipse surrounding the blob must have a strong, consistent orientation. The circles used as fiducials in Figure 14-7 would not be adequate since they do not have a strong elliptical orientation. If more than one fiducial is used, the fiducial does not need an identifiable orientation.

If other features on the board make it impossible to identify uniquely a fiducial with a blob finder, you may be able to use prototype models to locate fiducials. To use prototypes you must first create a prototype model of the fiducial using the camera that will eventually locate the fiducial. You then use a prototype finder to locate the fiducial.

A computed frame record requires the use of other vision tools (usually finder or general tools) to generate a frame based on what the vision tools find in the field of view. This method can be used effectively with one fiducial. Remember, however, all located fiducials must use the same vision operation. You cannot define different vision operations for different fiducial locations.

Some general requirements of the vision record used to locate a fiducial are:

- The vision operation must be able to identify the fiducial.
- The same vision operation is repeated at each fiducial location. You cannot specify a different vision operation depending on which fiducial you are looking for.

¹ Vision records "return" classes of information. One class of information is a vision reference frame that is relative to the camera field of view. Records that return frame results are said to be in the frame class. See the *VisionWare User's Guide*.

- If only one fiducial is used, the tool must be able to identify the orientation of the fiducial.
- ☒ **TopLevel** must be selected in the vision record or the record will not be shown in the pick lists displayed when completing arguments for a REFINE_FRAME statement.

The vision tool record requires camera and picture records with the following characteristics:

Camera Record

Camera records are described in Appendix C of the *VisionWare User's Guide*. For a REFINE_FRAME statement, the camera used must be an arm-mounted camera, and it must be calibrated. (See section 14.2 for details on calibrating an arm-mounted camera.)

Picture Record

Picture records are described in Appendix C of the *VisionWare User's Guide*. For a REFINE_FRAME statement, the picture record has the following requirements (to change these options, select ☒ **Use it** from the motion group and choose **Setup** :

- ☒ **Object moved by robot** should not be selected.
- Select ☒ **Vision target location**. This selection causes the robot to move the camera, rather than the gripper, over the location. The offset from the corner of the camera field of view to the center of the robot tool flange that was created when the camera was calibrated allows the robot to move to a vision target location rather than a robot location.
- Leave the PATH name data box blank.
- Entering a settling time will ensure that the robot stops completely and will dampen camera vibration. Higher robot speeds, heavier payloads, and smaller fields of view require longer settling times.

Optional Records for REFINE_FRAME

The following records are used by the optional clauses in the REFINE_FRAME statement.

Approach Path

This path will be used when the robot moves to the first picture-taking location.

Default Frame

If a default frame is used, the vision target location(s) will be relative to this frame. Based on where the fiducials are found, the database reference frame will be updated and used as the reference frame for the placement locations on the assembly. The default frame is not updated as each assembly is built.

Remember:

If a default frame is used, the vision operations will always be performed in the same place (since this frame is not updated by a REFINE_FRAME statement). The default frame must be close enough to the actual board reference frame for the fiducial marks being located to lie within the vision field of view.

If a default frame is not used, the assembly database frame must have an initial value that is close enough for the vision operation to find the fiducial mark. Since the database reference frame is updated each time a REFINE_FRAME statement is executed, the picture-taking locations for the next REFINE_FRAME statement will be

based on the updated database reference frame (the picture-taking locations will not always be in the same place).

Depart Path

This path will be used when the robot moves from the last picture-taking location to its next destination.

Tool

Specifies a tool transformation (see “Tool Transformations” on page 102) to be used with this statement. This tool will override the current tool.

Additional Considerations

When you build the records needed for a REFINE_FRAME statement:

1. Create all locations with the proper tool transformation in place.
2. The first `-- frame --` record specified in the REFINE_FRAME statement should also be specified as the “Named Frame” reference frame for all the location parameters in this statement.¹
3. Define the fiducial marks to be relative to the first reference frame specified in the REFINE_FRAME statement. Select ☒ **Named** in the location record and enter the frame name.
4. If a tool transformation is used, camera calibration must be performed with the proper tool transformation in place. If a prototype finder locates the fiducial, it must use the same camera that created the prototype model.
5. When debugging the vision operation, observe the database reference frame value before and after the vision operation. If it changes drastically, either the camera calibration is incorrect or parameters on the picture record are set up incorrectly.

¹ This frame must come from the frame database (not the vision database).

Using Conveyor Systems With Vision **15**

15.1 Vision With Indexing and Tracking Conveyors	216
15.2 Vision and Indexing Conveyors	219
15.3 Vision-Based Tracking Conveyor Options	222
Vision and Tracking Conveyors	224
15.4 Advanced Conveyor Operations	224

15.1 Vision With Indexing and Tracking Conveyors

With indexing conveyors, the MotionWare vision option allows you to define vision operations that will locate all objects in the field of view each time the conveyor stops. The robot can pick up all objects detected by the vision system. The vision system allows parts to be placed on the conveyor at random spacing and in random orientation.

With tracking conveyors, the MotionWare vision option allows you to define vision operations that will regularly take pictures of a moving belt and pass to the robot queue for pickup the locations of any objects found.

MotionWare uses vision data from a fixed-mount camera to return locations where parts are to be acquired or placed. The instructions in this chapter generally refer to removing objects from a conveyor. However, the same strategies can be used to either load a conveyor or build an assembly on a conveyor. In either case, the vision operation used must be able to positively identify the location where a part is to be acquired or placed.

Three kinds of information are needed to locate and acquire a part for indexing conveyors:

1. The vision transformation that relates the corner of the field of view to the robot base. This transformation is created during camera calibration.
2. The distance from the corner of the field of view to the object in the field of view is required; it is generated by the vision system for each new part that is seen.
3. The height and orientation of the gripper when the part is acquired. This data is created during the first execution of a sequence using the conveyor belt. Figure 15-1 shows the transformation created during camera calibration and the transformation created each time new parts move into the field of view and a picture is taken.

An arm-mounted camera cannot be used with the conveyor manager.

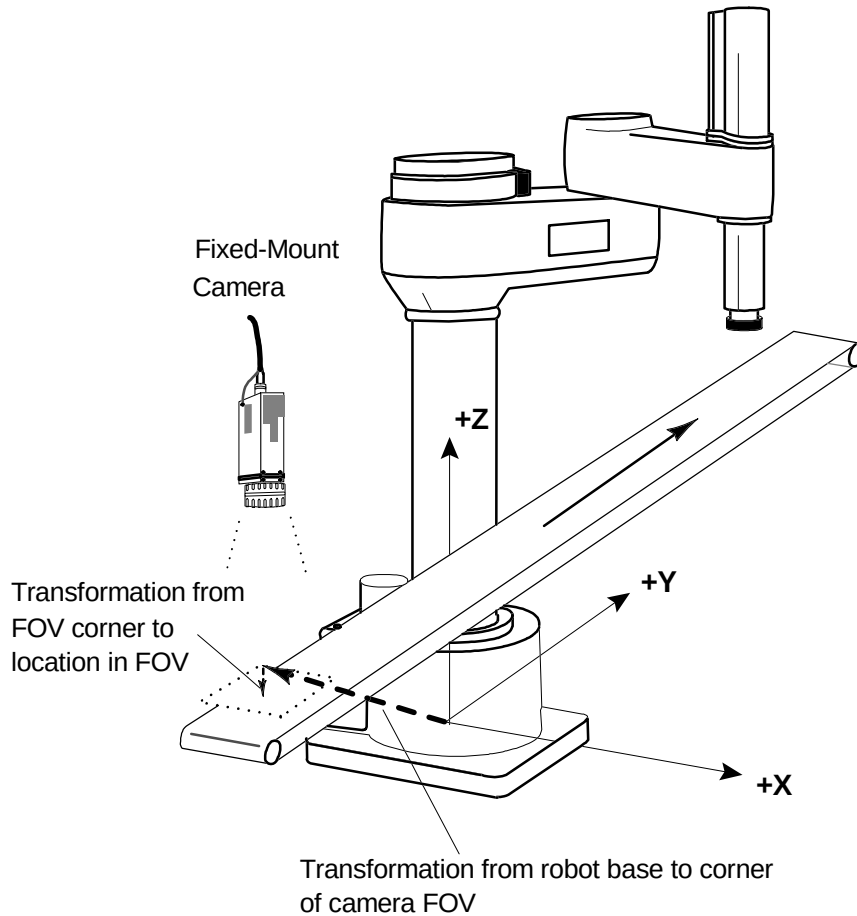


Figure 15-1
Conveyors and Vision

Tracking conveyors require the same information as indexing conveyors, plus the following additional information:

A transformation that relates the belt direction of travel to the world coordinate system, and the relationship between the belt encoder data and the distance traveled by the belt. This information is generated by a belt calibration procedure that can be performed from within MotionWare.

MotionWare uses the belt calibration data to create a “moving” reference frame that is constantly updated as the belt moves. This reference frame is calculated when a picture is taken and an object is located. Based on belt-calibration data, this reference frame is continuously updated until the robot has tracked and acquired the part. Then a new moving reference frame is calculated and updated based on the next object located by the vision system.

The *VisionWare User's Guide* describes the vision processes that can be used to identify object location and orientation. You must be familiar with the material in the *VisionWare User's Guide* before you can make effective use of vision with conveyor belts. Vision records are a special type of database record in that they have different *types* and generate different results based on what type of record they are. The types of records that are

important for conveyor operations are records that return a *frame* result. The vision record types that return frame results are listed in Table 15-1.

Table 15-1
Records That Return Frame Results (“Frame Class” Records)

Record Type	Frame Returned
Computed Frame	The computed frame.
Blob Finder	Origin is at blob centroid; orientation varies depending on selections made from the “Orientation” group in the blob-finder record.
Prototype Finder	Based on frame of trained prototype.
Arc Ruler	Origin is at first edge found; X-axis goes from arc center through first edge found.
Linear Ruler	Origin is at first edge; orientation is the same as the ruler tool.
Frame Pattern	See Chapter 9 of the <i>VisionWare User’s Guide</i> .
Point Finder	Origin is the found point; orientation is the same as the point finder reference frame.
Computed Point	Origin is the computed point; orientation is the same as the computed point reference frame.
Arc/circle Finder	Origin is the center of found arc; orientation is the same as the arc finder reference frame.
Computed Circle	Origin is the center of computed circle; orientation is the same as the computed circle reference frame.
Power Frame	See the <i>VisionWare User’s Guide</i> chapter of the AIM 3.1 Release Notes.
Refine Frame	See the <i>VisionWare User’s Guide</i> chapter of the AIM 3.1 Release Notes.

The vision record types most commonly used with conveyor belts are blob and prototype finders. Blob finders are used when all parts on the conveyor can be distinguished by their area alone and do not touch. Prototype finders are used when the part shape is required to distinguish parts, or when the parts are touching or overlapping. Computed frame records are used for parts whose orientation cannot be correctly calculated by a blob finder.

Before vision can be used with a conveyor, the cameras must be calibrated (see “Camera Calibration” on page 192). Camera calibration relates the size of the field of view to the size of objects in the field of view, and calculates the relationship between the field-of-view location and the robot base.

15.2 Vision and Indexing Conveyors

To create an indexing conveyor that uses vision to locate objects on the conveyor:

Edit ➡ Conveyor

The following screen is displayed:

Conveyor (mowsam)

Go Seek Edit Help

sample.idx.vis 09-Oct-96 09:11
1 of 1 Robot_1

Description
This is a sample indexing conveyor that uses vision to locate objects.

Wait Path: conv.wait

Continuous — Object Source —
☒ Yes ☐ No ☒ Sensor ☐ Vision ☐ External

Null or Indexing Conveyor

	X/y	Y/p	Z/r
Define	102.240	56.770	45.678
	0.000	0.000	0.000

Static Frame

Advance time: 10.00 sec
 Delay after stop: 1.00 sec
 Robot clearance: 10.00 mm [Here](#)
 Picture zone: 0.00 mm [Here](#)

Vision object spacing: 10.00 mm
☒ New picture for each object

Control Signals

Inputs: ☐ Enabled ☐ Ready ☐ Sensor
 Outputs: ☐ Advance

[Define](#)

Figure 15-2
Indexing Vision Conveyor

- Create a conveyor record as described in section “Indexing Conveyor Records” on page 142 and make the following changes:
 - Select **● Vision** as the Object Source.
 - In Vision object spacing, specify the minimum distance that must exist between two objects before the robot will attempt to acquire the second object. (We recommend you start with one half the minor axis of the part.)

- Make sure that all frame records that refer to this conveyor have the vision option selected.
- Select the ☒ **New picture for each object** option to force a new picture to be taken after each object is accessed. Only one object is placed in the queue. When the object is removed from the queue, a new picture is taken. When this option is selected, the conveyor cycles only when no object is found.

NOTE: For more complex programmed control of the conveyor, see the RESUME_CONVEYOR statement on page 161.

2. Ensure that the conveyor frame record is enabled.
3. Create a new record in the Frame database (**Edit ➔ Frame**) that has the following characteristics:

Select ☐ **Object Definition**

Select ☐ **Vision**

The screenshot shows the 'Frame (mowsam)' dialog box with the following settings:

- Go Seek Edit Help** menu bar.
- idx.vis.frame** in the name field, **1 of 1** below it.
- Fixed** and **Variable** radio buttons, with **Variable** selected.
- Date field: **09-Oct-96 09:20**.
- Object Definition** radio button selected.
- Device:** **Robot_1**.
- Description:** **This is a sample record for an indexing conveyor using vision.**
- Object Definition** section:
 - Conveyor:** **sample.idx.vis**
 - Non Vision**, **Vision**, and **External** radio buttons, with **Vision** selected.
 - Vision locate:** **locate**
 - Vision inspect:** (empty field)
 - Order:** (empty field)
 - Use object enable signal** checkbox is unchecked.
- Variable Status** section:
 - Object:** (empty field)
 - Result:** (empty field)
 - X:** (empty field)
 - State:** (empty field)
 - Code:** (empty field)
 - Y:** (empty field)
- At the bottom, a **Frame Value** field with a long text input area.

Figure 15-3
Frame Record for Indexing Conveyor With Vision

4. Create the vision operation that locates the objects in the field of view and returns a frame result representing their position.

Select the ☒ **TopLevel** option in the vision record. If multiple objects are to be located, make sure the record type chosen is a repeating record and the ☒ **Repeat** option is selected.

Use this record in the **Vision locate** field of the Frame record.

5. Create a location record and make it relative to this object frame. Select ☐ **Named** and enter the name of the reference frame, defined in step 3. Leave the location value blank. It must be taught during walk-thru training.

6. Display the Task Control Panel and start the conveyor server and vision server tasks. Choose the **Proceed** push button. The conveyor will index once and a picture will be taken (ensure that there is an object in the field of view).
7. Proceed through the sequence until an **Undefined value--edit** message is displayed for the location record. Choose **Edit**. When the location record is displayed, teach the location of the object that is in the field of view.

Values that are recorded in the location data box represent the offset and rotation from the frame origin (returned by the vision system) that the gripper should use when picking up a part. Since the vision system can locate objects only in the X/Y plane, the Z value recorded in this step will be used to determine the tool flange height for all part acquires.

15.3 Vision-Based Tracking Conveyor Options

To create a vision-based tracking conveyor:

Edit ➡ Conveyor

The following screen is displayed:

Conveyor "Global"

Go Seek Edit Help

sample.belt 11-Oct-96 11:17

1 of 1 Robot_1

Description

This is a sample vision based conveyor.
Its frame rec. must use vision option.

Wait Path: conv.wait

Continuous Object Source

☐ Yes ☒ No ☐ Sensor ☒ Vision ☐ External

Continuous Conveyor Parameters, Belt 1

Upstream limit: 0.25 mm Here

Dynamic wait line: 13.61 mm Here

Belt stop line: 1177.40 mm Here

☐ Only stop busy objects

Pickup limit: 1145.00 mm Here

Downstream limit: 1211.60 mm Here

Robot clearance: 0.00 mm Here

Picture distance: 20.00 mm

Overlap count: 2

Vision object spacing: 10.00 mm

Export: ☐ New ☐ Ignored ☐ Done Queue:

Belt Info Load Belt

Calibrate Store Belt

Control Signals

Inputs Outputs

☐ Enabled ☐ Ready ☐ Advance

☐ Sensor

Define

Figure 15-4
Vision-Based Tracking Conveyor

1. Create a conveyor frame record as described in "Tracking Conveyor Frame Records" on page 146. Make the following changes:
 - a. Select ☒ **Vision** as the Object Source.

- b. If an object may be seen in more than one picture (the same area of the belt is seen in more than one picture), the robot will attempt to access that object more than once.

In Vision object spacing, specify the minimum distance that must exist between two objects before the robot will attempt to acquire the second object. (We recommend you start with one half the minor axis of the part.)

In the overlap count data box, specify the maximum number of overlapping pictures expected during belt operation. (If this number is too small, "phantom" objects will be queued.)

- c. Make sure that all frame records that refer to this conveyor have the vision options selected.
 - d. Whenever possible, set up the workcell so the robot does not interfere with the camera field of view. Objects may be missed if they pass under the camera while the robot is in the way.
2. Create a frame record that has the following characteristics (this record must be in a module frame database and not the global frame database):

Select ☒ **Object Definition**

Select ☒ **Vision**

The screenshot shows the 'Frame (mowsam)' dialog box with the following settings:

- File:** sample.conv.vis
- Mode:** Fixed (selected), Variable
- Date/Time:** 09-Oct-96 09:20
- Object Definition:** (selected)
- Device:** Robot_1
- Description:** This is a sample record for a tracking conveyor using vision.
- Object Definition Section:**
 - Conveyor:** sample.belt
 - Use object enable signal:** (unchecked)
 - Non Vision:** (selected)
 - Vision locate:** locate
 - Vision inspect:** (empty)
 - External:** (selected)
 - Order:** (empty)
- Variable Status Section:**
 - Object:** (empty)
 - Result:** (empty)
 - X:** (empty)
 - State:** (empty)
 - Code:** (empty)
 - Y:** (empty)
 - X Y Z y p r:** (empty)
 - Frame Value:** (empty)

Figure 15-5
Frame Record for Tracking Conveyor With Vision

3. Create the vision operation that locates the objects in the field of view and returns a frame result representing their position.

Select the ☒ **TopLevel** option in the vision record. If multiple objects are to be located, ensure that the record type chosen is a repeating record and that the ☒ **Repeat** option is selected.

In the picture record associated with the vision record, you will normally select the ☒ **Strobe** option. The strobe option is used not only to trigger a strobe light, but also to record the belt location when a picture is taken. This will provide maximum accuracy during high-speed operations. Use this record in the Vision-locate field of the frame record defined in step 2.

4. Create a location record and make it relative to this frame.

Select ☒ **Named** and enter the name of the frame. Leave the location value blank. It must be taught during walk-thru training.

5. Turn the conveyor belt off and place one of the objects being acquired in the field of view.
6. Ensure that the conveyor is enabled.
7. Display the Task Control Panel and start the conveyor server and vision server tasks. Choose the **Proceed** push button. The conveyor will index once and a picture will be taken (ensure that there is an object in the field of view).

Run the conveyor belt until the object can be reached by the robot (if necessary).

8. Proceed through the sequence until an `Undefined value--edit` message is displayed for the location record.

Choose **Edit**.

Teach the location of the object. The values that are recorded in the location data box represent the offset and rotation from the frame origin (returned by the vision system) that the gripper should use when picking up a part. Since the vision system can locate objects only in the X/Y plane, the Z value recorded in this step will be used to determine the quill height for all part acquires.

9. Turn the conveyor on and proceed through the rest of the sequence.

Vision and Tracking Conveyors

When vision is used with a tracking conveyor, pictures are taken and MotionWare accesses objects based on the vision results. This allows MotionWare to access objects that are randomly placed and oriented on the belt. Before a vision guided tracking conveyor can be used:

- The camera must be calibrated. See Chapter 14.
- Moving belts will generally require a strobe light to acquire clear images.
- The belt must be calibrated. See Section 12.10.

15.4 Advanced Conveyor Operations

The optional arguments in a `GET_FRAME` statement allow you to select from multiple queued objects. The required frame argument must be an Object or Variable frame record.

The statement will return the first object it finds, subject to the following constraints:

- The `DISABLE` argument is a mask value that determines which objects are accessed. A bit in this integer value corresponds to each of the eight optional frame arguments. If a bit is set, that object will be ignored.

- The OUTPUT argument is a similar bit field that indicates which objects were found. During multiple access you can use the OUTPUT value to modify the DISABLE value until all desired objects have been accessed.

NOTE: If more than one object is desired, it can be used for more than one of the optional frame arguments.

This MATCH_CODE argument to the GET_FRAME statement allows you to specify which objects should be accessed. The code is a 16-bit integer. The RELEASE_FRAME statement sets this code. Using the two statements allows you to pass information from an upstream robot that is exporting objects to a downstream robot.

MotionWare Utilities

16

16.1 The File Manager	228
File Manager Options	228
16.2 The Backup and Restore Utility	231
The Backup Menu Page	231
Backup Definitions	231
Creating a Backup Definition	232
Restoring Files	234
Restore Options	236

16.1 The File Manager

The File Manager allows you to copy, delete, rename, and move files and directories.

To access the File Manager:

Utilities ➡ File Manager

The following screen is displayed:

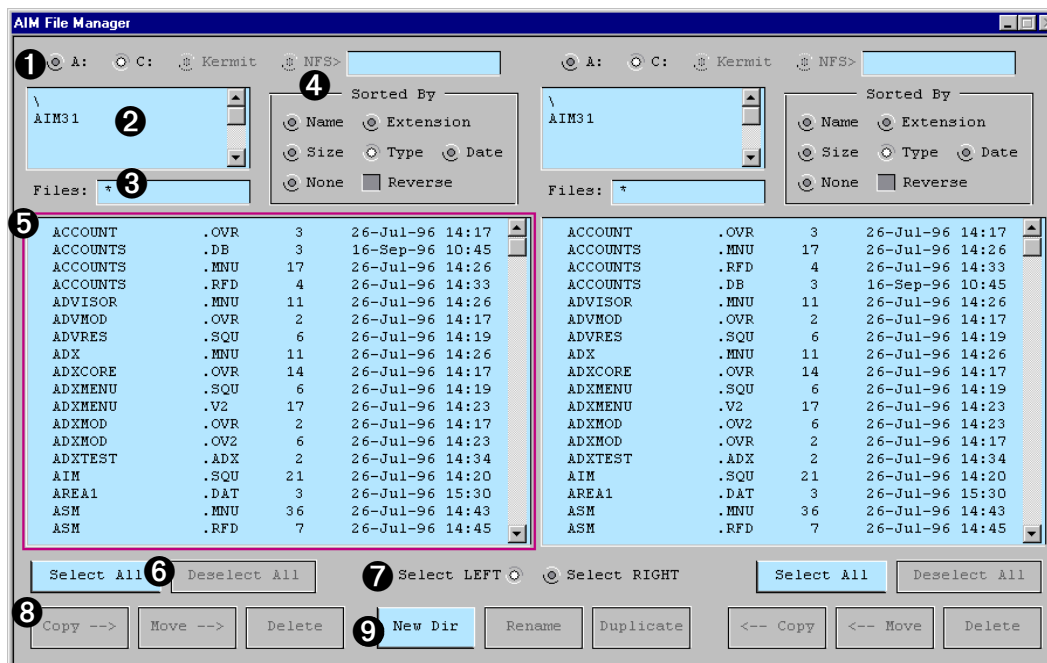


Figure 16-1
File Manager Window

The File Manager screen contains two scrolling list windows. Each window displays all files and directories in the corresponding selected directory.

File Manager Options

- 1 Select A:, C:, Kermit, or NFS> as the source device for this half of the window.¹ Before NFS> can be selected, the account name must be entered in the data box next to NFS>.²

Any device can be reread by clicking on the NFS> radio button (even if it is currently selected).

- 2 This window shows the directory path for item 5. Double-click any directory in the path to display the files and directories in that directory.²

¹ The Kermit device is available only if your controller has been configured to use a serial port to access another computer running a Kermit server. Then the Kermit device provides access to files and directories on that computer. See the *V+ Language User's Guide* for details.

NFS is available only to systems equipped with the optional AdeptNet hardware and software. NFS allows you to access files and directories on a remote computer on a network. See the *AdeptNet User's Guide* for details.

² This information is stored for the current AIM session and will be redisplayed each time the File Manager is opened.

③ This string can be used to limit the files displayed in item ⑤. (This string does not affect the directories displayed in item ⑤). For example, the string "*.v2" would limit the display to the files with that extension.¹

④ These options specify the order in which the files and directories in item ⑤ are displayed:¹

- ☐ **Name** sorts the files alphabetically based on the file name
- ☐ **Extension** sorts the files alphabetically based on the file extension
- ☐ **Size** sorts the files based on their size
- ☐ **Type** sorts the files based on the type attribute (directory, protected, etc.)
- ☐ **Date** sorts the files based on the creation/update date
- ☐ **None** does not sort the files (shows files as they are stored on disk)

Select ☒ **Reverse** to sort files in reverse order. (Has no effect on None.)

⑤ This area shows, for the device selected in ① and the directory path shown in ②, all of the directories present and all of the files present that match the specification entered in ③. This is the area where you select files or directories to copy, delete, rename, duplicate, or move. Click once on a file or directory to select it. To deselect an entry, click the entry again.

Selected files and directories appear with a ✓ to the left. Unselected files are black, unselected directories are blue, and selected files and directories are red. Selected system files are orange, and unselected system files are gray. Directories are identified by a "D" in the attribute column (immediately after the file size). Protected files are marked with a "P" and system files are marked with an "S".

To move one level down in the directory structure, double-click the desired directory entry.

⑥ Choose **Select All** to select all the files and directories shown in item ⑤.

Choose **Deselect All** to deselect all the files and directories that are selected.

⑦ You can operate on files and directories from only one side of the File Manager. These radio buttons indicate which side is the active side. The active side also is outlined in red. You can select a side by clicking one of the radio buttons or any entry in item ⑤, or choose **Select All**.

⑧ Choose **Copy -->** to make copies of the selected files and place them in the directory shown on the opposite side of the window.

Choose **Move -->** to move the selected files to the directory shown in the opposite side of the window.

Choose **Delete** to delete the selected files.

¹ This information is stored for the current AIM session and will be redisplayed each time the File Manager is opened.

NOTE: Files cannot be restored after they are deleted (except if they had previously been saved with the backup utility).

- ⑨ Choose **New Dir** to create a new directory in the active window. Enter the name for the new directory when prompted.

Choose **Rename** to rename the highlighted file or directory. A window will prompt for the new name.

Choose **Duplicate** to create a copy of the selected file in the same directory. (A window will prompt you for a name for the copied file.)

NOTE: **Rename** and **Duplicate** are available only when a single line is selected in item ⑨. **Duplicate** cannot be used on a directory entry. The window that prompts for the new name can be dragged aside in order to get an unobstructed view of the main window.

16.2 The Backup and Restore Utility

The backup utility allows you to create backup files of your AIM and V+ files. Backup files can be used to restore files in the event of a drive failure, file corruption, or accidental deletion.

The Backup Menu Page

The backup utility is accessed by:

Utilities ➡ Backup

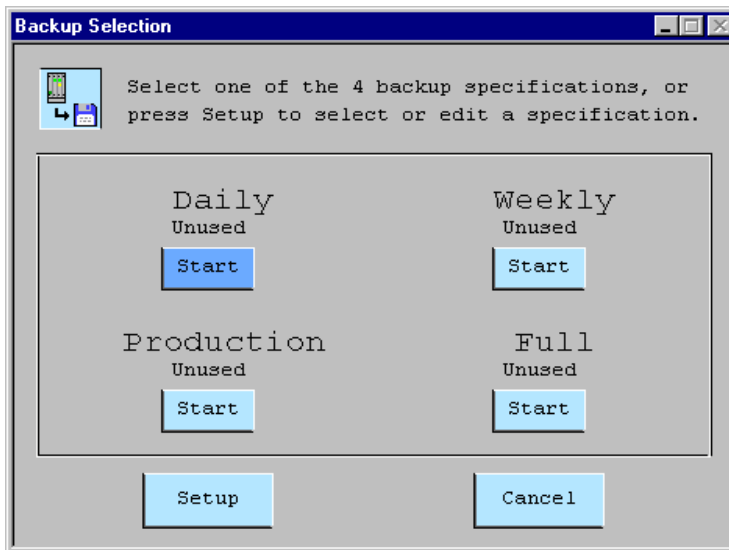


Figure 16-2
Backup Selection

This menu page has buttons for up to four backup definitions. Each button shows the name of a backup definition and the date of the last backup. Choosing a **Start** button will begin the backup using the indicated backup definition.

You can select which backup definitions appear on this page when you create or edit a backup definition (see item ⑤ on page 234).

To start a backup using a definition that is not assigned to one of the buttons, to modify a backup definition, or to create a new backup definition, choose **Setup** to display the window titled “List of Backup Definitions” on page 232.

Backup Definitions

Before you create a backup file, you need to create a backup definition that specifies which files to back up, the destination for those files, and the file(s) identification.

To create a backup definition:

Utilities ➡ Backup ➡ Setup

The following window is displayed:

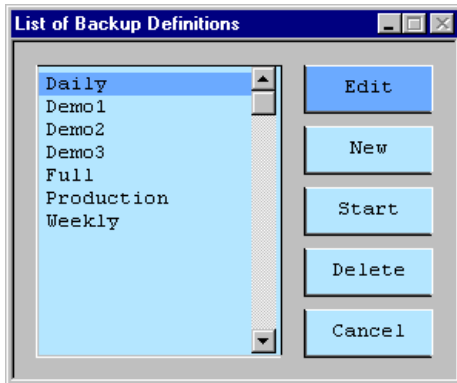


Figure 16-3
List of Backup Definitions

Choose **Edit** to change characteristics of the selected backup definition. (Double-clicking the name of a backup definition has the same effect as choosing this button.)

Choose **New** to create a new backup definition (see the next section).

Choose **Start** to begin creating a backup file using the highlighted backup definition. A status window will display the state of the backup operation and allow you to pause or abort the operation.

Choose **Delete** to delete the selected backup definition.

Choose **Cancel** to exit the backup utility. (The Exit function key and clicking the close-window icon have the same effect.)

Creating a Backup Definition

When you choose either **Edit** or **New** from the List of Backup Definitions window, this additional window is displayed:

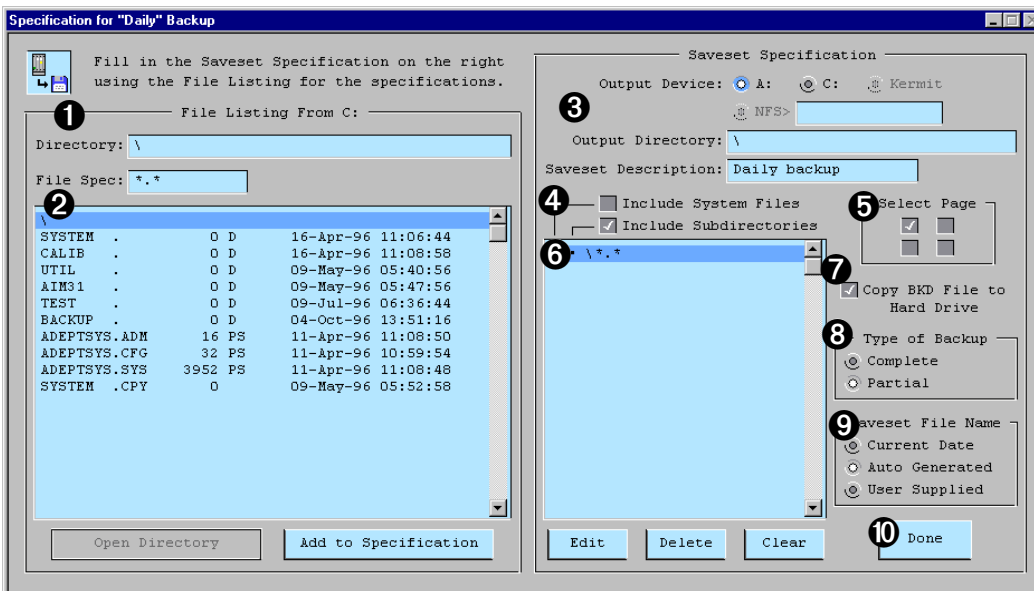


Figure 16-4
Backup Definition

- ❶ This area allows you to specify the source of the files to back up and which file groups to add to the backup definition.

Indicate the path to the source files. This should be an absolute path (see the *V+ Language User's Guide*). You can change the path by directly entering a path; double-clicking any directory (indicated by a "D" after the file size in item ❷); or selecting a directory and choosing **Open Directory**. (Double-click the ".." line to move up the directory tree one level.)

The File specification data box allows you to restrict the files displayed in the source file window.

To select all files, type:

.

To select a group of files (all database files, for example) type:

*.db

See the *V+ Language User's Guide* for details on using wild cards.

- ❷ This window shows the files that can be added to the backup definition from the specified path, matching the file specification. You can either add all of the files shown or select individual files.

To add all files, select the "." line (single period at the top of the list) and choose **Add to Specification**.

In the example shown, all database files have been selected ("*.db" in the file specification). To place all these files in the backup definition, select the "." line and choose **Add to Specification**. The path and file specification will be added to the Saveset list (item ❸). Note that all the individual file names will not be added.

To add selected files from the source list, select each individual file and then choose **Add to Specification**.

- ❸ Specify where the backup files will be stored:
- Select the desired output device. If you are backing up to an NFS drive, type a mount name and then select **NFS>**.¹
 - Indicate the directory specification for the backup files.
 - You can also enter a description of this backup definition.
- ❹ Select ☒ **Include system files** to include in the backup list (item ❹) any V+ system files that are included in the selected entry or directories within that entry.

Select ☒ **Include subdirectories** to include in the backup list all subdirectories of the selected directory entry.

¹ The Kermit device is available only if your controller has been configured to use a serial port to access another computer running a Kermit server. Then the Kermit device provides access to files and directories on that computer. See the *V+ Language User's Guide* for details.

NFS is available only to systems equipped with the optional AdeptNet hardware and software. NFS allows you to access files and directories on a remote computer on a network. See the *AdeptNet User's Guide* for details.

NOTE: These selections are made for each entry in the backup list, not for all of the entries in the list. Highlight each entry in the list and select the desired options for that entry.

- 5 Select one of the check boxes if you want this definition to appear as one of the standard buttons on the Backup Selection window (see Figure 16-2). If another backup definition has already been assigned to that button, a window will warn you and then allow you to reassign the button or keep the current assignment.
- 6 This list shows all the specifications that will be included in the backup. Full names are shown for files added individually. A wild card specification is shown for files added as a group. A specification can be edited or deleted by selecting the line and then choosing one of the buttons at the bottom of the window.
- 7 There are two (or more) files associated with a backup: a backup definition file and the actual backup files that contain all the selected files.

Select ☒ **Copy BKD File** to make a copy of the definition file in the backup directory of the C: drive as well as in the Output directory. (The backup directory is defined in an initialization record in the BASEINI initialization file.) This local copy of the definition file is not needed, but its presence will simplify any subsequent Restore operation for the backup.

- 8 Select ☐ **Complete** to back up all files regardless of whether they have been backed up previously.

Select ☐ **Partial** to back up only files that have changed since they were stored in a backup and files that have never been stored in a backup.

- 9 Select ☐ **Current date** to use the current date as the file name (in the form mmddyy).

Select ☐ **Auto generated** to have a random value assigned as the file name.

Select ☐ **User supplied** to assign your own file name (when the backup is started).

- 10 Choose **Edit** to manually edit the selected file specification. (Double-clicking a specification has the same effect as choosing this button.)

Choose **Delete** to remove the selected file specification from the backup list.

Choose **Clear** to remove all items from the backup list.

Choose **Done** to exit the backup specification editor. (The Exit function key and clicking the close-window icon have the same effect.)

Restoring Files

The file-restore utility reinstalls files contained in a backup file that was previously created with the backup utility.

To start the restore process:

Utilities ➡ Restore

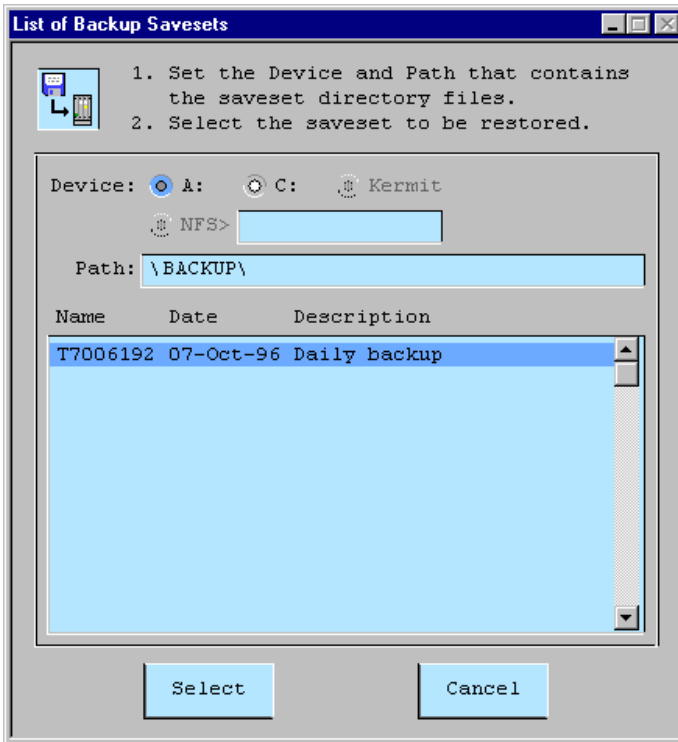


Figure 16-5
List of Backup Savesets

This window shows backup directory files at the indicated device and path. To prepare to restore the files, highlight the desired saveset and choose **Select** or double-click the saveset. The restore menu page is displayed.

The restore utility can be exited by choosing **Cancel**, pressing the Exit function key, or clicking the close-window icon.

Restore Options

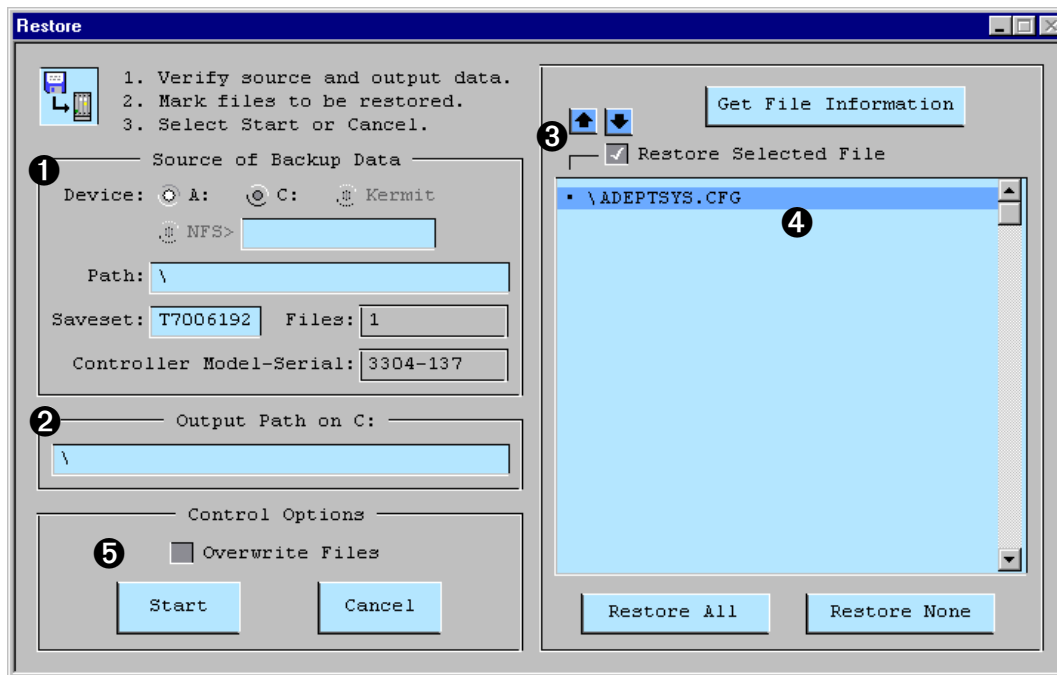


Figure 16-6
Restore Menu Page

- ❶ This area shows information about the backup selected from the backup savesets list. You can change the device, path, and saveset name in this window. However, it is usually easier to cancel this window and make changes in the List of Backup Savesets window.
- ❷ Indicates the target path to receive restored files.

NOTE: The path will be added to the path stored in the backup definition file. Thus, to restore files to their original location, the path should be “\” (or blank).
- ❸ A check mark indicates that the file will be restored. To select/deselect specific files, select the file and then click the check box. The up and down arrow buttons can be used to move through the list.

Choose **Restore All** to select all of the files for restore.

Choose **Restore None** to deselect all of the files.
- ❹ This area indicates the files stored in the backup file. The files marked with a check mark will be restored. Either double-click a file name or choose **Get File Information** to see basic information on that file.
- ❺ Select ☒ **Overwrite files** to have the utility automatically overwrite files of the same name. Choose **Start** to begin restoring files.

Once a restore operation has started, a window will show the status of the restore and give you the option of pausing or aborting the operation.

Installing MotionWare **A**

- A.1 Installing the Software 238
 - Create a Subdirectory for the AIM Files 238
 - Copy the AIM Files to the Hard Drive 238
 - Upgrade the Operating System 239
- A.2 Initialization Databases 240
 - Initialization Options 240
- A.3 Autostarting MotionWare 244

A.1 Installing the Software

To install MotionWare on your hard drive:¹

1. Create a subdirectory for MotionWare.
2. Copy the MotionWare files to the hard drive.
3. Upgrade the operating system (if necessary). See sections following for specific information.

Create a Subdirectory for the AIM Files

Turn on the controller and perform the steps below when the startup procedures have completed.

At the controller keyboard, type:

fdirectory/c c:\aim

A new subdirectory will be created. To make that directory the default directory, type:

default = c:\aim

To verify that the directory was properly created and is the default directory, type:

default

The bottom line of the monitor window will display:

DISK>C:\aim

If this line is not displayed, repeat the steps in this section.

Copy the AIM Files to the Hard Drive

Use the V⁺ utility program DISKCOPY to copy the AIM baseline system floppy disks to your hard drive. Complete instructions for using the utility program are contained in the *Instructions for Adept Utility Programs* supplied with your V⁺ user manuals.²

The following section describes the procedure to copy the diskette files to your hard drive.

1. Turn on your controller.
2. After the startup procedures have completed, make AIM the default directory by typing:

default = c:\aim

3. Run the DISKCOPY program.
Place the Adept Utility Disk in the A drive and type:

load a:diskcopy

4. After the program file is loaded, type:

execute 1 a.diskcopy

5. Remove the utility disk. Insert an AIM disk into the A drive.

¹ If you have purchased a new system, AIM may already be installed.

² If you are using MotionWare with the vision extension, ALL vision and VisionWare files must be copied to the same directory that contains the MotionWare files. Also, if you are using add-in applications such as PCB, all add-in application files must be in the same directory that contains the MotionWare files.

6. From the menu that is presented, choose:
 4 => COPY multiple FILES
7. When prompted What is the INPUT disk?, type:
 a
8. When prompted What is the OUTPUT disk?, type:
 c
9. When prompted Do you want to verify copy operations?, type:
 y
10. When prompted Enter spec of file(s) to copy, type:
 ,
11. When prompted Enter output subdirectory,
 Press the ENTER key.
12. When prompted Do you want existing files automatically superseded?, type:
 y
13. When prompted Copy this file, type:
 g
14. Repeat steps 10-13 for the remaining AIM disks.¹
15. After the last disk has been copied, when prompted Enter spec of file(s) to copy,
 Press the ENTER key.
16. Choose exit.

Upgrade the Operating System

If MotionWare is being delivered with a new system, the correct operating system is already installed. If you are adding MotionWare to an existing system, you may be required to upgrade your operating system.

To determine if you need to upgrade your operating system, turn on the controller and type:

id

The ID command shows the system configuration. The first number in the software line is the operating system version. It must be 11.2 or higher. Otherwise, you must upgrade to the current version.

To upgrade your operating system:

1. Turn on the controller and let it load the existing operating system.
2. Insert the Adept Utility Disk (delivered with the existing system software) in the A drive and type the following command to load the file CONFIG_C.V2:²

¹ If you are not using the guided vision option, you do not need to copy the Vision or VisionWare disks.

² Or use the correct drive and path specification if the file is stored on the hard drive.

load a:config_c.v2

When loading is complete, you can remove the Utility Disk from the disk drive.

3. Start execution of the program by typing the command:

execute a.config_c

(Or use **execute 1 a.config_c** to execute the program as task #1.)

4. See *Instructions for Adept Utility Programs* for the procedure to use CONFIG_C to install a new V⁺ system.

NOTE: If a floppy diskette is modified, the write-protect slider on the back of the diskette must be in the unprotected position. That is, it must be in the position that closes the hole through the diskette jacket. Ensure that you move the slider back to the protected position after the operation is completed.

Do not alter the original V⁺ system diskette supplied by Adept. Instead, make a copy of the distribution diskette (with this program during the process of installation, or with the DISKCOPY utility). Then modify the configuration information on the copy.

5. The configuration program can be deleted from the system memory by typing the following commands (type **kill 1** in place of **kill** if the program was executed as task #1):

```
kill
deletem a.config_c
```

A.2 Initialization Databases

MotionWare has several initialization databases that control MotionWare at startup. To change an initialization value:

Setup ➡ Initialization Data ➡ *select an initialization file*

Seek ➡ Index ➡ *select an initialization record*

Initialization Options

The following tables summarize the initialization options available to MotionWare.

Table A-1

Options in the Baseline File (BASEINI.DB)

Record Name	Use
abort "Are you sure?" enable	Enables/disables the "Are you sure?" verification when the Abort button is pressed on the Debug Control Panel.
access level, initial	Sets the system access level at startup. (Do not change this record unless you have created a user account with an access level 4.)
access level, V ⁺ monitor	Sets the access level required to display the monitor window.

Table A-1
Options in the Baseline File (BASEINI.DB) (Continued)

Record Name	Use
advisor enable	Enables/disables the AIM Advisor. The AIM Advisor provides interactive help for various AIM programming and setup tasks.
autostart module name	The name of the module to load and select when autostarting.
autostart sequence name	The name of the sequence to execute when autostarting.
backup copy directory	The name of the subdirectory to use with the Backup/Restore utility program.
cycle repeat, initial	Sets the Repeat option on the Task Control Panel.
database auto-save enable	Enables/disables auto-saving database records whenever a database editing page is exited.
DDE cache size	Specifies the maximum number of entries that can be placed in the Dynamic Data Exchange (DDE) cache.
DDE enable	Enable/disable the DDE driver.
error/message log queue size	Used by AIM customizers only.
error/message log queue task	The number of the task to handle the error/message queue.
GEM enable	Enables/disables the Generic Equipment Model (GEM) communications option. Requires purchase and installation of the Adept GEM software option.
I/O server enable	Enables/disables the I/O server software.
Kermit directory file start	Specifies the position of the first character of the file name of a directory record returned by Kermit to the File Manager.
language definition, English	Specifies English as one of the available languages (other languages may be added by system customizers).
language selection	Specifies which language to display if multiple languages have been implemented.
module default directory	The default directory for newly created resource modules.
screen blanking time interval	Sets the idle time before the screen will go blank.
signal for operator attention	The digital I/O signal that should be set whenever the *operator attention* message is generated.
stack size, sequence nesting	The maximum depth of nested sequences that can be called from a single task.
statements - baseline	The name of the baseline statement database.

Table A-1
Options in the Baseline File (BASEINI.DB) (Continued)

Record Name	Use
statements - task control	The name of the task control statement database.
task number, concurrent number	Specifies the total number of tasks searched when looking for a free task for a concurrent AIM process.
task number, first concurrent	Specifies the first task searched when looking for a free task for a concurrent AIM process.
user name, initial	The name of the user to be logged on at startup.

Table A-2
Options in the Robot Module File (ROBINI.DB)

Record Name	Use
conveyor manager enable	Specifies whether to load the conveyor manager.
display digits for transforms	Select the decimal precision for display of transformation values (does not affect computational accuracy).
flexible feedware, enable	Enables/disables the Flexible FeedWare option. Requires purchase and installation of the Flexible FeedWare hardware and software option.
force sensing option enable	Specifies whether to load the force sensing option.
high-accuracy position enable	Enables/disables use of the optional Adept High-accuracy Positioning System (HPS). HPS must be installed on the system and an HPS map must have been created and stored.
robot approach minimum	Sets the absolute minimum height for robot moves that involve a dynamically generated reference frame. This option ensures that the robot tool tip will not move below the indicated height.
robot speed, initial	Sets the speed displayed on the Operator Control Panel the first time that the control panel is displayed after system startup. Also used as the autostart speed.
robot speed, maximum	Sets the highest speed that the user will be allowed to set on the Operator Control Panel.

Table A-3

Options in the MotionWare Initialization File (MOWINI.DB)

Record Name	Use
control sequence module	The name of the module to use when looking for menu button control sequences. Normally set to "mowctl". Defines sequences used by the Master Control Panel.
import module – mowctl	Defines a module archive file for import into the current module database during startup.
import module – mowsam	Defines a module archive file for import into the current module database during startup.
import module – vcalmow	Defines a module archive file for import into the current module database during startup.
master control panel file name	The name of the menu file that will be accessed by the Master Control Panel pull-down.
master control panel page name	The name of the menu page that will be accessed by the Master Control Panel pull-down.
statements - MotionWare	The MotionWare statements database.

Table A-4

Options in the User File (USERINI.DB)

Record Name	Use
error database, additional	This record allows you to create your own error/message database and have it loaded along with the standard AIM error database.
overlay, additional	The name of a custom V+ program to be loaded as an overlay during startup. The file name must be of the form *.ovr and the first program in the file must be the same as the file name.
statements, additional	The name of a custom statement database to be merged with the standard AIM statements.

If you are using MotionWare with the optional vision extension, you will have the vision initialization database (VISINI). You also will have a vision record-type initialization file (MOWRTINI.DB) that contains basic information on the vision record types available with the vision option. Unless you are creating your own vision record type or want to specifically disable some existing record type, you will not need to use this file. These files are discussed in the *VisionWare User's Guide*.

If other AIM application or add-in modules are loaded, their initialization databases are described in documentation accompanying those programs.

A.3 Autostarting MotionWare

MotionWare can be automatically started at system power-on. Before starting MotionWare, complete these steps:

1. Before the controller is turned on:

Set the autostart position on the configuration DIP switch to autostart. The location of the configuration DIP switch on the SIO module is detailed in the *Adept MV Controller User's Guide*.

2. Before starting MotionWare:

Make sure the root directory of the C drive is the current directory; type:

default disk = c:

Copy the MotionWare autostart file from the distribution diskette to the root directory and give the file the correct name.

For a single robot system, type:¹

fcopy auto.v2 = a: mowauto.v2

For a two robot system, type:

fcopy auto.v2 = a: mowauto2.v2

3. After starting MotionWare, open the BASEINI.DB initialization file:

Setup ➔ Initialization Data ➔ db1 clk "baseini.db"

4. Select the record autostart module name.

If autostart module name is not displayed:

Seek ➔ Index

- a. Select autostart module name from the displayed list.
- b. Select **Go To**

Type the name of the resource module you want loaded and selected at startup.

5. Select the record named autostart sequence name.

If autostart sequence name is not displayed:

Seek ➔ Index

- a. Select autostart sequence name from the displayed list.
- b. Select **Go To**

Type the name of the sequence you want executed at startup (in most cases this will be a control sequence that performs all the work required to load and start an application sequence).

The BASEINI.DB initialization file also contains many other initialization preferences that allow you to customize how MotionWare starts. These include changing the vision-window display mode, enabling of runtime graphics, displaying the number of cycles to complete, displaying the user name for the initial logged-on user, and enabling of results logging.

¹ The MOWAUTO file contains a program that will set the default disk to C:\AIM\. MotionWare must have been copied to this directory or the program **auto** must be changed to reflect the directory that holds the MotionWare files.

Menu Bars and Quick Keys

B

B.1 Main Menu Bar	246
B.2 Record Editing Menu Bar and Quick Keys	248
B.3 Vision Window Menu Bar	251
B.4 Sequence Editing Menu Bar and Quick Keys	253

B.1 Main Menu Bar

These menus are displayed whenever MotionWare is running. Several options are used only with the AIM system, which underlies MotionWare.

Execute	
Master Control Panel	<i>Displays the master control panel</i>
Task Control Panel	<i>Displays the task control panel</i>
Stop All Tasks	<i>Stops all user tasks (started from Master or Task Control Panel)</i>

Edit	
Sequence	<i>Edits a sequence</i>
Variables	<i>Edits or creates records in the Variable database</i>
Vision	<i>Edits or creates vision records (if the vision extension is loaded)</i>
Camera	<i>Edits or creates camera records</i>
Conveyor	<i>Edits or creates records in the Conveyor database</i>
Frame	<i>Edits or creates records in the Frame database</i>
Path	<i>Edits or creates records in the Path database</i>
Tool	<i>Edits or creates records in the Tool database</i>
Locations	<i>Edits or creates records in the Location database</i>

Module	
Select/Edit	<i>Selects or Edits a resource module</i>

Show	
Statement Tree	<i>Displays a graphic tree of the individual statements in a sequence</i>
AIM Status Window	<i>Displays the AIM status window</i>
Vision Results	<i>Displays the Inspection Results menu page (if the vision extension loaded)</i>
Live Video	<i>Selects camera and displays live video output for that camera</i>
Runtime Graphics	<i>Selects display of runtime graphics (if the vision extension loaded)</i>

Conveyor Queues	<i>Shows the status of the conveyor queues</i>
Raw Conveyor	<i>Shows the raw conveyor data</i>

IO	
Digital Signals	<i>Displays/sets digital I/O signals</i>
Log Results	<i>Sets options for logging vision results</i>

Setup	
Initialization Data	<i>Accesses the various initialization databases</i>
Calibrate Camera	<i>Calibrates a specific camera</i>
Display/Change Tool	<i>Accesses the robot tool utilities</i>
Calibrate Conveyor	<i>Calibrates a defined conveyor</i>
Cross Calibrate Conveyor	<i>Cross calibrates two conveyors already calibrated to the same conveyor belt</i>

Utilities	
Save All DBs	<i>Saves changes made to any databases in memory</i>
Set Access Level	<i>Sets the current user's access level</i>
File Manager	<i>Displays the AIM file manager</i>
Backup	<i>Creates a backup definition</i>
Restore	<i>Restores the system files from a backup definition</i>
DBM Utilities	<i>Enters the database creation and editing program</i>
Edit menu	<i>Enters the menu editing program</i>
V+ Developer	<i>Opens the V+ developer option (see the AIM Customizer's Guide for these three options)</i>
Import/Export Windows	<i>Imports/Exports a vision image or Exports a TIFF file of a menu page</i>

Profiler	<i>Displays the system CPU usage profiler</i>
Export ADX Data	<i>Exports a database to an ADX file format</i>
Import ADX Data	<i>Imports an ADX formatted file into a database</i>

Special	
Sample Menu	<i>Displays a sample menu for use with menu editing program</i>
Edit Advisor	<i>Edits the Advisor database</i>
Edit Error Messages	<i>Edits the Error database</i>
Edit Help File	<i>Edits Help databases</i>
Edit Init Data	<i>Edits the AIM initialization databases</i>
Edit Raw Sequence	<i>Edits the Sequence database</i>
Edit Statements	<i>Creates or edits AIM statements</i>
Edit User Accounts	<i>Creates new users, edits or deletes existing users</i>
Link All Databases	<i>Reexecutes the linking process that is performed when a sequence is executed</i>
Trace Enable/Disable	<i>Enables/disables output to the Monitor window</i>
System Shutdown	<i>Exits AIM and saves all modified databases</i>

Help	
Advisor	<i>Browses the Advisor database</i>
Error Messages	<i>Browses the Error Message database</i>
System ID	<i>Displays the system ID window</i>

B.2 Record Editing Menu Bar and Quick Keys

This menu is displayed whenever you are editing database records. The quick keys that correspond to different menu options are shown in parentheses ("SHIFT+" indicates that the shift key should be pressed and held down while a specified quick key is pressed).

Go	
Goto	<i>Goes to the selected record (F3)</i>

Select	<i>Displays a pick list of records appropriate to the selected field (F5)</i>
Tree	<i>Displays the record tree showing other records associated with this record</i>
Back	<i>Goes back to the previously open record</i>
Retrieve	<i>Returns the record name of the open record to the previously open record (SHIFT+F3)</i>
Exit	<i>Exits record editing mode, does not permanently save new or altered records to the disk (F7)</i>

Seek	
Index	<i>Displays a pick list of all records in the current database</i>
First	<i>Goes to the first record in the database (Home)</i>
Previous	<i>Goes to the previous record in the database (Page Up)</i>
Next	<i>Goes to the next record in the database (Page Down)</i>
Last	<i>Goes to the last record in the database (End)</i>
Find	<i>Enters a field value, searches the database of the selected field for the value (F7)</i>
Repeat Find	<i>Repeats the last find operation (F8)</i>

Edit	
Undo Field	<i>Performs an undo (toggle between old value and new value) on un-entered changes made from the keyboard (F6)</i>
New Record	<i>Creates a new database record (F2)</i>
Cut Record	<i>Cuts the current record to the copy buffer (SHIFT+F9)</i>
Copy Record	<i>Copies the current record to the copy buffer (F9)</i>
Paste Record	<i>Pastes the record in the copy buffer to the current database (F10)</i>
Default Device	<i>Sets the device to be used for newly created records</i>
Tweek Motions	<i>Changes motion parameters for groups of records</i>

Help	
Field Information	<i>Displays the help screen for the currently selected menu item (F1)</i>
Page Information	<i>Displays help screen for the current menu page (SHIFT+F1)</i>

Whenever the record editing menu is displayed, the following quick keys will be active.

Table B-1
Record Editing Quick Keys

Key Combination	Effect
HELP (F1)	Displays help for the current menu page or the currently selected menu page item.
NEW (F2)	Creates a new record in the current database and goes to that record.
SAVE (SHIFT+F2)	Displays the "Save All Modified Databases" window.
GO TO (F3)	Goes to the record when a source field is selected.
RETRIEVE (SHIFT+F3)	Returns the record name of the open record to the previously open record.
EXIT (F4)	Closes the currently open menu page and returns to the next open page (if any).
SHIFT+F4	Closes all open menu pages.
DISPLAY (F5)	Displays a pick list of records appropriate to the selected field when a source field is selected.
UNDO (F6)	Performs an undo (toggle between old value and new value) on the most recent changes entered from the keyboard (only if the changes have not been entered by pressing ENTER or moving to another field).
REDRAW (SHIFT+F6)	Forces a redraw of the screen (updates digital I/O references).
FIND (F7)	Displays the Find window.
REPEAT (F8)	Repeats the last executed find operation.
COPY (F9)	Copies the currently displayed record.
CUT (SHIFT+F9)	Cuts the currently displayed record and places it in the copy buffer.
PASTE (F10)	Pastes the record currently in the copy buffer.
TAB	Selects the next menu item. The next menu item is the physical item to the right of the currently selected menu item, or the left-most item below the currently selected menu item. If the last menu item is selected, the first menu item will be selected.
SHIFT+TAB	Selects successive menu items in the reverse order of Tab.

Table B-1
Record Editing Quick Keys (Continued)

Key Combination	Effect
SHIFT+↑, ↓, →, ←	Selects successive menu items in a top/bottom or left/right order.
ENTER	Causes the system to accept the data box entry if a data box is selected. If a button is selected, has the same effect as if the button is clicked on with the mouse pointer.
PAGE UP	Goes to the previous record in the database.
PAGE DOWN	Goes to the next record in the database.
HOME	Goes to the first record in the database.
END	Goes to the last record in the database.
CTRL+DELETE	Deletes the value in a selected field.

B.3 Vision Window Menu Bar

This menu bar is displayed in the vision window whenever a vision record is being edited.

The vision window and menu bar are displayed by:

Edit ➔ Vision

Image	
New Picture(s)	<i>Acquires required images for the current operation (redraws any graphics)</i>
Show Edge Image	<i>Shows the grayscale edge image</i>
Live Grayscale	<i>Shows the actual camera output (in grayscale mode), when selected</i>
Live Binary	<i>Shows the actual camera output (in binary mode), when selected</i>
Grayscale Image	<i>Shows the image currently stored in the grayscale frame buffer, when selected</i>
Binary Image	<i>Shows the image currently stored in the binary frame buffer, when selected</i>
Image Lock	<i>Takes a new picture only when New Picture(s) is selected (useful when developing applications using strobe lights)</i>

Show	
Vision Tools	<i>Displays the graphics for all currently defined vision tools, when selected</i>
Points	<i>Displays graphics for currently defined point class recs, when selected</i>
Lines	<i>Displays graphics for currently defined line class recs, when selected</i>
Circles	<i>Displays graphics for currently defined circle class recs, when selected</i>

Vision Frames	<i>Displays graphics for currently defined frame class recs, when selected</i>
----------------------	--

Delete All	
Delete all but pictures and models	<i>Deletes all vision records except camera, picture, prototype model, and font records</i>
Delete all vision records	<i>Deletes all vision records (in the currently open database)</i>

Goto	
Top Level List	<i>Displays a list of top-level vision inspections</i>
Top of tree	<i>Redraws the currently displayed tree from the top</i>

Options	
Vision tool timer	<i>Hides or shows the timer in the lower left corner of vision window</i>
Show edge points	<i>Enables or disables the display of points used for finder tools</i>

Zoom	
Zoom In	<i>Zooms in on the vision window</i>
Zoom Out	<i>Zooms out on the vision window</i>
Zoom Popup...	<i>Displays the window that lets you see the zoomed area before zooming</i>
Back to normal	<i>Displays the unzoomed vision image</i>

The following menu items are available only when using AdeptWindows PC:

Resolution	
Full	<i>Displays the vision image at full resolution</i>
Half	<i>Displays the vision image at half resolution</i>
Quarter	<i>Displays the vision image at quarter resolution</i>
Focus	<i>Displays full resolution at the center of the vision image</i>

B.4 Sequence Editing Menu Bar and Quick Keys

This menu bar is displayed whenever you are editing a sequence. The sequence editor and menu bar is displayed by:

Edit ➔ Sequence

File	
Export File	<i>Exports the sequence to an ASCII file</i>
Exit	<i>Exits the sequence editor—changes saved to RAM only (F4)</i>
Go	
Sequence Index	<i>Displays the select/create/delete sequence window</i>
Goto Field Record	<i>Goes to the selected record</i>
Display Field Index	<i>Displays a pick list of records appropriate to the selected argument (F5)</i>
Move	
First Line	<i>Moves cursor to the first line of the sequence (CTRL+Home)</i>
Last Line	<i>Moves cursor to the last line of the sequence (CTRL+End)</i>
Previous Page	<i>Shows the previous screen of statement lines (Page Up)</i>
Next Page	<i>Shows the next screen of statement lines (Page Down)</i>
Find	
Repeat Find	<i>Repeats the last find operation (F8)</i>
Repeat Change	<i>Repeats the last change operation (SHIFT+F8)</i>
Find Setup	<i>Specifies a string to find in the current sequence (F7)</i>
Change Setup	<i>Specifies a string to find in the current sequence and another string to replace the found string with (SHIFT+F7)</i>
Edit	
Undo Field	<i>Performs an undo on recent (unentered) changes made from the keyboard (F6)</i>
Delete Field	<i>Deletes the selected record or statement name (CTRL+U)</i>
Disable	<i>Enables or Disables the current line (Esc)</i>

Expand	<i>Displays or hides the optional arguments in the current statement line (F11)</i>
Append Line	<i>Appends a new line to the end of the sequence (F2)</i>
Insert Line	<i>Inserts a new statement line above the current cursor location (CTRL+Insert)</i>
Delete Line	<i>Deletes the current statement line — not placed in copy buffer (CTRL+Delete)</i>
Copy Line	<i>Copies the current line to the copy buffer (F9)</i>
Cut Line	<i>Cuts current line and places it in copy buffer (SHIFT+F9)</i>
Paste Line	<i>Pastes the first line in the copy buffer above the current cursor location (F10)</i>
Paste All	<i>Pastes the entire copy buffer above the current cursor location (SHIFT+F10)</i>

Table B-2
Sequence Editing Quick Keys

Key Combination	Effect
NEW (F2)	Appends a new statement line to the end of the sequence.
SAVE (SHIFT+F2)	Displays the Save All Modified Databases window.
GO TO (F3)	Displays a pick list of statements when <code>--statement--</code> is selected. When an uncompleted statement argument is selected, displays a pick list of appropriate records. When a completed statement argument is selected, goes to the record.
RETRIEVE (SHIFT+F3)	Returns to the sequence editor after a “Go to”.
EXIT (F4)	Exits the sequence editor.
DISPLAY (F5)	Displays a pick list for the selected statement or argument.
UNDO (F6)	Performs an undo operation (toggle between old value and new value) on recent changes entered from the keyboard (unless the ENTER key has been pressed or another menu item has been selected).
FIND (F7)	Displays the Find window.
CHANGE (SHIFT+F7)	Displays the Change window.
REPEAT (F8)	Repeats the last executed find operation.
SHIFT+F8	Repeats the last executed change operation.
COPY (F9)	Copies the currently selected statement line to the copy buffer.
CUT (SHIFT+F9)	Cuts the currently selected statement line and places it in the copy buffer.
PASTE (F10)	Pastes the most recently cut or copied line above the currently selected statement line.

Table B-2
Sequence Editing Quick Keys (Continued)

Key Combination	Effect
PASTE ALL (SHIFT+F10)	Copies all lines in the copy buffer above the currently selected statement line.
EDIT (F11)	Expands or contracts the selected statement.
Double-click on the statement line	Expands the currently selected statement.
Double-click on the statement number	Adds or removes a comment marker.
SHIFT + ↑, ↓, →, ←	Selects successive statement arguments in a top-to-bottom or left-to-right order.
→, ←	Moves the typing cursor within a statement name or argument.
ENTER	Moves to the first argument in the next statement.
DELETE	Deletes character at the typing cursor.
BACKSPACE	Deletes the character to the left of the typing cursor.
SHIFT+INSERT	Adds a statement above the current statement line.
SHIFT+DELETE	Deletes the current statement line (not placed in copy buffer).
HOME	Goes to the first line of the current screen.
END	Goes to the last line of the current screen.
SHIFT+HOME	Goes to the first line of the sequence.
SHIFT+END	Goes to the last line of the sequence.
PAGE UP	Displays the previous screen of statements.
PAGE DOWN	Displays the next screen of statements.
ESC	Adds or removes comment marker(s).
CTRL+U	Blanks the selected statement or argument name.

A

- Access level 23, 246, 247
- Accounts
 - creating user 24
- Active/inactive
 - menu options 20
- Adept Bulletin Board 7
- Adept HyperDrive User's Guide* 78
- AdeptForce
 - operator interface 124
- AdeptVision AGS-GV User's Guide* 3
- AIM CAD Data Translator User's Guide* 4
- AIM MotionWare manual set 3
- AIM resource modules 40
 - loading and selecting 40
- AIM tasks 68
- Approach heights
 - specifying 32
 - with locations 75
- Arguments
 - completing 54
 - types 60
- Arm configuration
 - setting 33
- ASCII files
 - writing databases to 44
 - writing sequences to 44
- Auto loading modules 42
- AUTO.V2 244
- Autostart 244

B

- Backing up files 231
- Backup/Restore utility 231–236
- BASEINI.DB
 - Initialization database 240
- Baseline file 240
- Belt
 - calibration utility 154
 - limits 140
 - status window 155
 - window 140
- BELT_CAL.V2 154
- BFD (backup file definition) 231
- Bulletin Board, Adept 7
- Buttons, pressing 18

C

- Calibrating a conveyor 152
- Calibration
 - arm-mounted camera 200
 - fixed-mount camera 197
 - loading conveyor belt calibration 154
 - options 196
- CALL statement 178
- Camera
 - calibration 192
 - fixed-mount 192
 - robot-mounted 192
- Camera record
 - with REFINE_FRAME 213
- Camera resource 45
- Changing sequence arguments 59
- Choosing items 18
- Clauses, optional 53
- Clicking 14
- Commenting statements 54
- Completing arguments 54
- Conditional statements
 - ELSE 179
 - END 179
 - EXIT_LOOP 182
 - IF 178
 - NEXT_LOOP 182
 - REPEAT 181
 - WHILE 181
- Continuous path motion 32
- Control panel speed and robot speed 31
- Control panels
 - master 70
 - task 64
- Control sequences 50
- Conveyor belt
 - position 155
 - scale factor 155
 - velocity 155
- Conveyor belt calibration
 - loading from BELTxx.DAT files 154
- Conveyor resource 45
- Conveyor statements 168
- Conveyors
 - advanced operation 224
 - and pallets 144
 - and strobe signal 220, 224
 - and vision 216

- belt calibration utility 154
- belt status window 155
- calibrating a tracking 152
- indexing 142
- indexing vs. tracking 140
- object 141
- reference frame 140
- terminology 140
- tracking 146
- Coordinate system 28
- Copying
 - files 228
 - modules 42
 - records 21
 - statement lines 55
- Creating
 - a sequence 50
 - directories 228
 - modules 42
- Current tool 104
- Cursor 20
 - keys 20
- Cutting records 21
- Cutting statement lines
 - sequence editor 55
- D**
- Databases
 - and MOVE statement 170
 - force 36, 126
 - frame 36, 171
 - global 47
 - importing/exporting ASCII files 44
 - location 35
 - path 36
 - tool 36, 171
 - variable 37, 84
 - vision and camera 37
- Debugging sequences 66
- Deleting files 228
- Deleting statements 54
- Depart height
 - specifying 32
- Depart heights
 - with locations 75
- Device, pointing 13
- Digital force sensors 120
 - configuring V+ for 120
- Digital I/O 106–109
 - displaying signal status 106
 - indexing conveyor 143
 - negative signals 108
 - numbering 107
 - statements 109, 114, 115, 116, 168
 - viewing states of 106
- Digital output
 - setting state of 106
- Directories
 - duplicating 228
- Directory
 - creating 228
 - displaying file directory 228
 - duplicating 228
 - for resource module 42
 - renaming 228
- DISKCOPY 238
- Double clicking
 - retrieving records 23
- Dragging 14
- DRIVE 173
- Dual-vision systems 47
- Dwell time 77
- E**
- Editing
 - a sequence 52–71
 - locations 74
 - modules 42
 - records 20
 - sequences
 - during execution 71
- ELSE 179
- END 179
- Entry locations
 - paths 94
- Executing
 - a sequence from a sequence 178
 - MotionWare 12
- Exit locations
 - path 94
- EXIT_LOOP 182
- Exiting record editing 22
- Exporting
 - databases 44
 - sequences 44
- F**
- F4 key 22
- File Manager 228–230
- Files
 - backing up 231
 - copying 228
 - deleting 228
 - displaying directory of 228
 - moving 228
 - renaming 228
 - restoring 231

- Finding
 - and changing text
 - sequence editor 59
 - records 22
 - sequence arguments 59
- FOR statement 180
- Force 119
 - database 126
 - example application 132
 - records 127
 - resource 45
 - signature 125
- Force Process Records 127
- Force sensing
 - guarded mode 121
 - modes 121
 - protect mode 123
- Force Sensing Module 120
- Force Sensor Status and Control Display 124
- Force Signature Display Page 125
- Frame
 - class records 218
 - database 100
 - with MOVE 171, 172
 - record
 - with REFINE_FRAME 212
 - resource 45
- Function keys
 - in sequence editor 56
- G**
- Global databases 47
 - and modules 47
- H**
- Highlighting 20
- HyperDrive option 78
 - and optimized moves 78
- I**
- I/O Server 114
 - example sequence 117
 - installing 114
 - protocol routines 114
 - statements 114
- IF 178
- Importing
 - sequences 44
- Importing databases 44
- Index
 - sequence 50
- Indexing
 - conveyor 142
 - with vision 219–221
 - vs. tracking conveyors 140
- Initialization databases 240
 - BASEINI.DB 240
 - MOWINI.DB 243
 - ROBINI.DB 242
 - USERINI.DB 243
- Inspection
 - arm-mounted camera 206
 - fixed-mount camera 204
- Installing VisionWare 238
- IO 110
 - command 106
 - statement 110
- IO_CLOSE 115
- IO_LIST 111
 - statement 111
- IO_LIST_VAR 111
 - Statement 111
- IO_OPEN 115
- IO_READ 115
- IO_WRITE 116
- J**
- Joint-interpolated motion 31
- L**
- Left configuration 33
- LOAD
 - statement 185
- Loading
 - modules 41
 - modules from a sequence 185
 - MotionWare 12
- Location
 - defined 35
 - record
 - creating 74
 - reference frames 77
 - resource 45
 - values 28
- Location database
 - with MOVE 171
- Locations 28–30
 - approaching 32
 - departing 32
 - editing 74
 - pallet 80
 - pausing robot at 77
 - precision 32
 - teaching 30

- undefined 37
- Logging off of the system 25
- Logging on to the system 25
- Looping structures
 - FOR 180
 - REPEAT 181
 - WHILE 181
- M**
- Main menu
 - options 246
- Managing files 228
- Manual control pendant
 - teaching locations with 35
- Manual Control Pendant User's Guide* 35
- Manual conventions 16
- Manuals
 - related 3
- Master control panel 70
- MATCH_OBJECTS 161
 - Statement 161
- MCP, see Manual control pendant
- Menu bars 247–254
 - main menu 246, 247
 - main menu bar vs.
 - window menu bar 15
 - record editing 248
 - sequence editor 253
- Menu options
 - active/inactive 20
- Menu pages
 - standard features 15
 - using 15
- Menu selection 17
- Module
 - directory 42
 - resources
 - types of 45
- Module utilities 42
- Modules 40
 - and global databases 47
 - and sequences 50
 - auto-loading 42
 - copying 42
 - creating 42
 - deleting resources 44
 - editing 42
 - importing/exporting ASCII 44
 - loading 41
 - managing 42
 - resource 40
 - selecting 41
 - unloading 187
- Motion information 204
- MotionWare
 - databases 35
 - loading and executing 12
- MotionWare initialization file 243
- Mouse 13
- MOVE 169
- MOVE statement 37
- MOVE_REL statement 172
- Moves
 - optimized 78
- Moving files 228
- MOWAUTO.V2 244
- MOWIN1.DB
 - initialization database 243
- N**
- NEXT_LOOP 182
- O**
- Object
 - conveyor 141
- Operating system
 - upgrading 239
- Optimized moves 78, 79
- Optional clauses 53
- Options, statement tree 63
- P**
- Pallet
 - controlling update of 81
- Pallet options
 - changes to 81
- Pallet parameters 81
- Pallets 80–82
 - and conveyors 144
 - row and columns 81
 - updating indices 81
- Panic button 68
- Parameters, search 137
- Part
 - acquisition 133
 - insertion 134
- Password 24
- Pasting records 21
- Pasting statement lines
 - sequence editor 55
- Path
 - database
 - with MOVE 170
 - for resource directory 42
 - locations
 - deleting 92

- records
 - and picture records 205, 207
 - and REFINE_TOOL 209
- resource 45
- segments
 - editing 94
 - options 94
- Paths 90–95
 - creating 91, 94
 - defined 36
 - deleting 92
 - editing 91
 - entry 94
 - exit 94
- PAUSE_TASK statement 185
- Pausing a task 185
- Picture
 - locations 204
 - records 204
- Picture record
 - with REFINE_FRAME 213
- Pitch 29
- Pointing device 13
- Precision
 - location 32
- Prototypes
 - and camera calibration 196
- Q**
- Quick keys
 - record editing 250
 - sequence editor 253, 254
- R**
- Read only
 - option for sequence 50
- Record
 - classes
 - frame 218
 - editing
 - menu bar options 248
 - quick keys 250
 - finding 22
- Records
 - calibration 193
 - camera 213
 - copying 21
 - cutting 21
 - deleting 21
 - force 127
 - frame 212
 - other aids for creating 23
 - pasting 21
 - picture 213
 - selecting different 21
- Reference frame 98–101
 - computing from 3 points 100
 - creating with vision 210
 - defined 36
- Reference manuals 3
- REFINE_FRAME 210
- REFINE_TOOL 207
- Renaming directories 228
- Renaming files 228
- REPEAT 181
- Resource module 40
 - directory 42
 - options 42
- Restoring files 231, 234
- RESUME_CONVEYOR 161
 - Statement 161
- RESUME_TASK statement 186
- Resuming a task 186
- Retry statement 68
- RETURN statement 185
- Right configuration 33
- rn.fp.search 136
- rn.mw.standard 76
- ROBINI.DB
 - initialization database 242
- Robot
 - arm configuration 33
 - locations 28
 - motion
 - acceleration profile 32
 - speed 31
 - absolute 32
 - wait options 174
- Robot module file 242
- Robot motions
 - straight line vs. joint
 - interpolated 31
- Roll 29
- Rotational speed
 - relation to Cartesian
 - rotation speed 31
 - setting 31
- Routines
 - standard force process 131
- S**
- Scrolling 15
- Search parameters 137
- Searching for and replacing text
 - sequence editor 59
- SELECT_TASK statement 186

- SELECT_VAR_REC 87
- Selecting 20
 - a task 186
 - items 18
 - modules 41
- Sequence
 - aborting 68
 - and module utilities 50
 - argument types 60
 - associated module 50
 - calling another sequence 178
 - control statements 168, 185
 - copying/pasting statement lines 55
 - creating 50
 - debugging 66, 71
 - disk file name 50
 - editing 52–71
 - editing during execution 66
 - execution 64
 - finding/changing arguments 59
 - index 50
 - pausing 64, 67
 - read only 50
 - read/write options 50
 - removing statements 54
 - resource 45
 - selecting for edit 50
 - selecting for task 64
 - single step execution 66
 - single stepping 72
 - starting 64
- Sequence control
 - LOAD 185
 - PAUSE_TASK 185
 - RESUME_TASK 186
 - RETURN 185
 - SELECT_TASK 186
 - START_TASK 186
 - STOP_TASK 186
 - UNLOAD 187
- Sequence editor
 - menu bar options 253
 - quick keys 56, 253
 - short cuts 56
- Sequences
 - and resource modules 50
 - and statement tree 62
 - control 50
 - importing/exporting ASCII files 44
 - performing 18
 - single step execution 71
- SET_FRAME 173
- SET_TOOL 174
- SETS 183
- Settling time 77
- Single step execution 72
- Single step execution of sequence 71
- Single step sequence execution 66
- Software
 - installing 238
- Speed
 - setting 31
 - setting absolute 32
 - setting robot 66
 - units 31
- START_TASK Statement 186
- Starting
 - a task 186
 - the system 12
- Statement
 - argument types 60
 - components 52
 - lines
 - copying and pasting 55
 - retrying 68
 - selecting in sequence 52
 - START_TASK 186
- Statement tree 62
 - options 63
- Statements 168
 - ANY 182
 - CALL 178
 - CASE Structure 182
 - commenting 54
 - conveyor 168
 - digital I/O 168
 - DRIVE 173
 - ELSE 179
 - END 179
 - EXIT_LOOP 182
 - FOR 180
 - GOTO 183
 - IF 178
 - inspect 203
 - IO 110
 - IO_LIST 111
 - IO_LIST_VAR 111
 - LOAD 185
 - MATCH_OBJECTS 161
 - MESSAGE 176
 - MOVE 169
 - MOVE_REL 172
 - NEXT_LOOP 182
 - optional arguments 53
 - PAUSE_TASK 185
 - picture 204
 - REFINE_FRAME 210
 - REFINE_TOOL 207

- removing from sequence 54
 - removing temporarily from
 - a sequence 54
 - REPEAT 181
 - RESET_PALLET 176
 - RESUME_CONVEYOR 161
 - RESUME_TASK 186
 - RETURN 185
 - SELECT_TASK 186
 - SELECT_VAR_REC 87
 - sequence control 168
 - SET 183
 - SET_FRAME 173
 - SET_RESPONSE 176
 - SET_TOOL 174
 - SETS 183
 - START_TASK 186
 - STOP_ROBOT 174
 - STOP_TASK 186
 - UNLOAD 187
 - VALUE 182
 - WAIT_UNTIL_ROBOT 174
 - WAIT 187
 - WAIT_FOR 109, 110, 115, 116
 - WAIT_UNTIL 110
 - WAIT_VAR 187
 - WHILE 181
 - STOP_ROBOT statement 174
 - STOP_TASK statement 186
 - Stopping a robot
 - from a sequence 174
 - Straight-line motion 31
 - Strobe
 - and conveyors 220, 224
 - System access levels 23
 - Systems
 - dual-vision 47
- T**
- Task
 - STOPPING 186
 - Task control panel 64
 - Tasks
 - description 68
 - pausing 185
 - resuming 186
 - selecting 186
 - starting 64, 68, 186
 - type 68
 - Teaching locations with the
 - manual control pendant 35
 - Tool
 - database
 - with MOVE 171
 - resource 45
 - transformations 102
 - Tools 102
 - and statements 104
 - current 104
 - Tracking conveyor 146
 - with vision 216–224
 - Tracking vs. indexing conveyors 140
 - Tree, statement 62
 - Typing cursor 20
- U**
- Undefined locations 37
 - UNLOAD statement 187
 - Unloading modules 187
 - User accounts
 - adding 24
 - editing 24
 - User file 243
 - USERINI.DB
 - initialization database 243
 - Users
 - adding 24
 - Utilities
 - backup and restore 231
 - file manager 228
 - module 42
- V**
- Variable database 84
 - Variable resource 45
 - Virtual cameras 196
 - Vision records
 - with REFINE_FRAME 212
 - Vision resource 45
 - Vision window
 - menu bar options 251
 - VisionWare
 - autostarting 244
 - installing 238
 - VisionWare User's Guide 3, 192, 218
- W**
- WAIT 187
 - WAIT_FOR 109, 115, 116
 - WAIT_UNTIL 110
 - WAIT_UNTIL_ROBOT statement 174
 - WAIT_VAR 187
 - Walk-Thru training 66
 - WHILE 181
 - Window features 13

Windows 13
 closing 15
 opening 14
 selecting 14
 sizing 15
World coordinate system 28

Y

Yaw 29

Adept User's Manual Comment Form

We have provided this form to allow you to make comments about this manual, to point out any mistakes you may find, or to offer suggestions about information you want to see added to the manual. We review and revise user's manuals on a regular basis, and any comments or feedback you send us will be given serious consideration. Thank you for your input.

NAME _____ DATE _____

COMPANY _____

ADDRESS _____

PHONE _____

MANUAL TITLE: _____

PART NUMBER and REV level: _____

COMMENTS:

MAIL TO: Adept Technology, Inc.
Technical Publications Dept.
11133 Kenwood Rd.
Cincinnati, OH 45242

FAX: (513) 792-0274

